



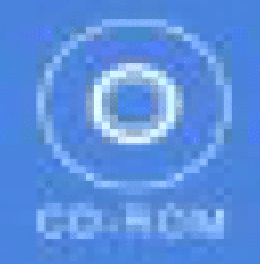
一本全面讲解Excel VBA精髓及应用的宝典秘籍

彻底理清Excel VBA的各种概念、开发技术及开发思想

Excel VBA 开发技术大全

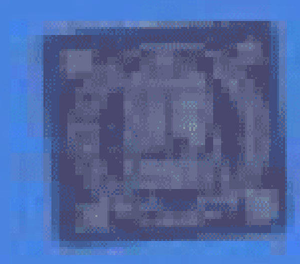
伍远高　等编著

- 大部分内容适用于Excel 2000/XP/2003/2007等多个版本
- 从零开始讲解，每个知识点都配典型实例讲解，可轻松上手
- 全面覆盖VBA基础、Excel对象模型、用户界面设计等内容
- 详细介绍如何使用外部数据，如用ADO访问数据库、处理文件等
- 深入讲解加载宏、操作VBE、调用Windows API、制作帮助系统等高级内容
- 提供301个实例、37个案例、214个技巧，可作为案头必备的查询手册



CD-ROM







清华大学出版社




原创经典，程序员典藏

一本全面讲解Excel VBA精髓及应用的宝典秘籍
彻底理清Excel VBA的各种概念、开发技术及开发思想

Excel VBA 开发技术大全

伍远高 等编著




清华大学出版社
北 京

内 容 简 介

Excel 2007 与以前版本相比，从操作界面到对象模型的变化都很大，例如，取消了菜单和工具栏，新增了功能区。本书在介绍通过 VBA 操作 Excel 对象的基础上，使用了大量篇幅介绍用 VBA 操作这些新增对象的方法。

本书共分 7 部分 31 章，分别介绍了 Excel 2007 开发平台概述、使用宏、Excel VBA 的开发环境、VBA 基础、程序控制结构、使用数组、使用过程、管理模块、处理字符串和日期、Excel 对象概述、使用 Application 对象、使用 Workbook 对象、使用 Worksheet 对象、使用 Range 对象、使用其他常用 Excel 对象、使用 Excel 内置对话框、创建自定义对话框、使用标准控件、使用 ActiveX 控件、使用 RibbonX、使用 CommandBars、控制其他 Office 程序、处理文件、使用 ADO 访问数据库、Excel 2007 与 Internet、使用 Excel 加载宏、使用类模块、操作 VBE、使用 Windows API、制作应用程序的帮助等内容。最后详细介绍了一个进销存管理系统的开发过程。

本书知识全面，结构由浅入深，每个知识点以实例代码进行介绍，使读者可快速入门。适合需要用 Excel 解决复杂问题，或准备利用 Excel VBA 技术开发 Excel 应用程序的读者，也适合大中专院校的学生阅读，还可作为 VBA 的培训教材。

图书在版编目（CIP）数据

Excel VBA 开发技术大全 / 伍远高编著. —北京：清华大学出版社，2009.2
ISBN 978-7-302-19214-5

Ⅰ. E…　Ⅱ. 伍…　Ⅲ. 电子表格系统，Excel 2007 – 程序设计　Ⅳ. TP391.13

中国版本图书馆 CIP 数据核字（2009）第 000381 号

责任编辑： 冯志强　赖　晓
责任校对： 徐俊伟
责任印制：
出版发行： 清华大学出版社　　**地　址：** 北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　编：** 100084
社 总 机： 010-62770175　　**邮　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　销： 全国新华书店
开　本： 185×260　**印　张：** 42　**字　数：** 1042 千字
（附光盘 1 张）
版　次： 2009 年 2 月第 1 版　**印　次：** 2009 年 2 月第 1 次印刷
印　数： 1～
定　价： 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：010-62770177 转 3103　产品编号：031017-01

前　　言

Excel 2007 是 Microsoft Office 2007 的组件之一，该软件主要用来对表格数据进行管理、分析、统计等，是办公人员最常用的软件之一。为了让 Excel 2007 发挥最大功效，可以借助于 VBA 开发各种电子表格应用程序。

使用 VBA 可以为 Excel 2007 应用程序提供新的功能或增强现有的功能，从而减少用户在 Excel 中的操作步骤，提高工作效率。如果要以 Excel 2007 为平台，使用 VBA 进行二次开发，则需要读者能熟练地操作 Excel 软件，并具有一定的程序设计能力。

本书特色

- 适用于多个版本：本书除第 20 章介绍 Excel 2007 新增 RibbonX 功能的内容外，其余章节的内容都可应用到 Excel 2000/XP/2003/2007 的各版本中。使用各版本的用户都可以通过本书学习 VBA 知识。
- 内容全面：市场上大多数介绍 VBA 类的书籍，都只是详细介绍了 Excel 对象模型的使用，没有程序设计基础的读者需要参考其他书籍来学习 VB 程序设计方面的知识。本书除了详细介绍 Excel 对象模型的使用外，还详细介绍了 VB 程序设计基础，使初学者通过本书就可学习到完整的 Excel VBA 程序设计的相关知识。
- 专业性强：本书除了介绍 VBA 相关知识外，还介绍了在 Excel 中调用 Windows API、使用 ADO 访问数据库、控制其他 Office 应用程序、使用类模块、制作帮助系统等应用程序开发中的高级内容，使读者开发的 Excel 应用程序更专业。
- 知识点和实例相结合：本书每个知识点都以实例代码来讲解。在本书最后以一个完整的进销存管理系统的开发过程为例，使读者能够通过实例进一步巩固前面各章所学的知识。

本书对 Excel 2007 的新增功能，以及实际开发应用程序中经常要用到、而其他书籍很少介绍的功能也进行了详细的介绍。例如：

- 使用 RibbonX，在第 20 章中详细介绍了使用 XML 自定义 Excel 2007 新增功能区的方法。
- 制作 COM 加载宏，在第 26 章中介绍了用 VB 开发 COM 加载宏的方法。
- 操作 VBE，在第 28 章中介绍了用 VBA 代码控制 Excel VBE 开发环境的方法。
- 制作帮助系统。在第 30 章中介绍了为 Excel 应用程序制作帮助文件的方法。

本书内容

本书共分 7 部分 31 章。

第 1 部分 Excel 2007 应用程序开发简介，包括 1～3 章，分别介绍了 Excel 开发平台概述、使用宏、Excel VBA 的开发环境等内容。

第 2 部分 VBA 基础知识，包括 4～9 章，分别介绍了 VBA 基础、程序控制结构、使用数组、使用过程、管理模块等程序设计的基本知识，第 9 章还详细介绍了字符串和日期的处理方法。

第 3 部分 掌握 Excel 对象模型，包括 10～15 章，详细介绍了 Excel 中常用对象的属性、方法和事件的使用方法（包括 Application 对象、Workbook 对象、Worksheet 对象、Range 对象、Chart 对象等常用对象的使用）。

第 4 部分 用户界面设计，包括 16～21 章，分别介绍了使用 Excel 内置对话框、创建自定义对话框、使用标准控件、使用 ActiveX 控件、使用 RibbonX 界面、使用 CommandBars 等内容。

第 5 部分 使用外部数据，包括 22～25 章，分别介绍了控制其他 Office 程序、处理文件、使用 ADO 访问数据库、Excel 2007 与 Internet 等内容。

第 6 部分 VBA 高级应用，包括 26～30 章，分别介绍了使用 Excel 加载宏、使用类模块、操作 VBE、使用 Windows API、制作应用程序的帮助等内容。

第 7 部分 综合应用程序设计，第 31 章为一个实例——进销存管理系统，本章详细介绍了该实例的开发过程，通过该实例的开发，进一步巩固前面各章所学的知识。

读者对象

本书要求读者已经能熟练使用 Excel 2007，并对 Excel 2007 的新增功能有一定的使用经验。在阅读本书前，读者至少已经掌握了以下的 Excel 操作技能：

- 格式化工作表；
- 命令单元格区域；
- 使用公式和函数；
- 创建图表；
- 管理工作簿；
- 管理工作表。

本书作者

本书由伍远高主笔编写。其他参与编写和资料整理的人员有王征、陈冠军、王石、程彩红、姜海英、邵毅、张路平、李臻、武勇、徐宁、刘玉珊、麻雪、赵建领、陈刚、齐晓宁、范永龙、姚志娟、赵盟、傅靖、李佳、徐磊、刘丹、肖冰、陈杰、王行恒、冯浩楠、

纪超、段桂东、颜盟盟、黄宝生、张珍珍、石淑珍、陈超、牛晓辉、刘聪、任潇、商斌、张双、于志华、李秀劲、李胜美、蔡文仙、杜阳阳、吴兴亮、陈水望、黄任桢、梅婷婷、皇波、白雪蛟、陈浩然、许程程、巩长宇、黄金亮、姜艳超、李军、李庆、彭志林、王志娟、武娜、尹成业等。在此一并表示感谢！

由于书稿内容涉及众多的计算机专业知识，且作者水平和学识有限，书中难免有疏漏之处，敬请读者朋友批评指正。

编著者

目　　录

第 1 部分　Excel 2007 应用程序开发简介

第 2 部分 VBA 基础知识

第 3 部分 掌握 Excel 对象模型

第 5 部分 使用外部数据

第 6 部分 VBA 高级应用

第 7 部分　综合应用程序设计

第 1 部分　Excel 2007 应用程序开发简介

Microsoft Excel 本身是一个流行的电子表格应用程序，但本书所谓的 Excel 应用程序是指以 Excel 为平台，在 Excel 环境中使用 VBA 为用户定制开发的应用程序，也就是通常所说的二次开发。通过二次开发，可根据用户的不同需求，定制出各种不同的应用环境。

第 1 章　Excel 2007 开发平台概述

Excel 作为专业的电子表格软件，被各行各业大量应用于表格制作、数据统计分析。Excel 内嵌 VBA 程序开发语言，从而为程序开发者提供了一个开发平台，可使有经验的用户对 Excel 进行定制。本章简单介绍用 Excel 2007 开发应用程序的基础知识和开发过程。

1.1　Excel 2007 新增功能

2006 年，微软推出了 Office 2007 套装软件，与以前版本相比，Office 2007 的变动非常大，采用了全新的界面。本节简单介绍 Excel 的发展过程及 Excel 2007 的新增功能。

1.1.1　Excel 版本简介

1993 年，微软正式推出了 Excel 5.0，该产品确立了微软在电子表格软件领域的重要地位。Excel 5.0 是一款里程碑级的软件，其所蕴涵的设计思想和先进技术深深地影响着所有的后继版本。Excel 5.0 运行于微软的视窗操作系统中，是 16 位的应用程序，运行在早期的 Windows 3x 操作系统中。

随着 Windows 95 操作系统的推出，微软在 Excel 5.0 的基础上不断推出新的 Excel 版本，例如，Excel 7.0、Excel 97、Excel 2000、Excel 2002、Excel 2003 等。

2006 年 11 月 30 日，微软正式推出 Office 2007（包括 Excel 2007 等）。在 Excel 2007 中，取消了传统的菜单加工具栏的操作方式，采用新的面向结果的用户界面。在新界面中，Excel 2007 提供了强大的工具和功能，可以通过应用主题和使用特定样式在工作表中快速设置数据格式。

1.1.2　Excel 2007 的特点

Excel 历来是需要分析信息的员工经常使用的一种工具。Excel 2007 成为功能强大的商业智能工具，可用于更安全地访问、分析及共享来自数据仓库和企业应用的信息。与以前版本相比，Excel 2007 主要改进和增强了以下几方面的功能。

- ❑ 基本电子表格功能。Excel 2007 可帮助人们更迅速地构建专业级别的电子表格，并且大大扩增了行列方面的处理能力，计算速度更快，改进了公式创建功能，并且增加了新的图库和样式模板。
- ❑ 商业智能分析功能。Excel 2007 可以连接到企业数据，并且保持电子表格和后台数

据源之间的持久连接。这样不仅便于利用最新信息来更新 Excel 工作表，而且能够在 Excel 里深入分析更详细的信息，查出异常和趋势。

- 增强的制图和打印输出。Excel 2007 利用了新的制图引擎，让人们能够制作专业外观的图表和图形。这些改进加上大大改善的打印效果，可以让人们共享重要报表里面的分析结果。
- “页面版式”视图。通过该视图可以让用户轻松查看每页工作表怎样打印以及分页符在哪里。用户给工作表添加了一行或者一列后，Excel 就会自动把文档的样式元素应用到新添的行或列上，而使用以前版本的 Excel，必须手动才能完成。
- 更醒目的条件格式。在 Excel 中可设置条件格式，让符合条件的单元格以用户提前设定的格式进行显示，例如，大于某值的单元格显示为红色等。在 Excel 2007 中，这一功能得到了大大的加强。
- 更方便的公式输入。在复杂的 Excel 工作表里，有时定义的公式涉及同一工作簿里的几个工作表，或不同工作簿中的工作表。在一个工作表里要想观察其相同工作簿下的其他工作表，或不同工作簿的其他工作表，是件很麻烦的事。在 Excel 2007 中，这一问题得到了解决，将要关注的单元格放在监视窗口中即可。不管当前编辑位置在哪里，监视窗口将始终显示。
- 更真实的打印预览。

1.1.3　Excel 2007 的界面

启动 Excel 2007 后，将出现如图 1-1 所示的界面。Excel 2007 的窗口主要包括标题栏、功能区、表格操作区、状态栏等部分。

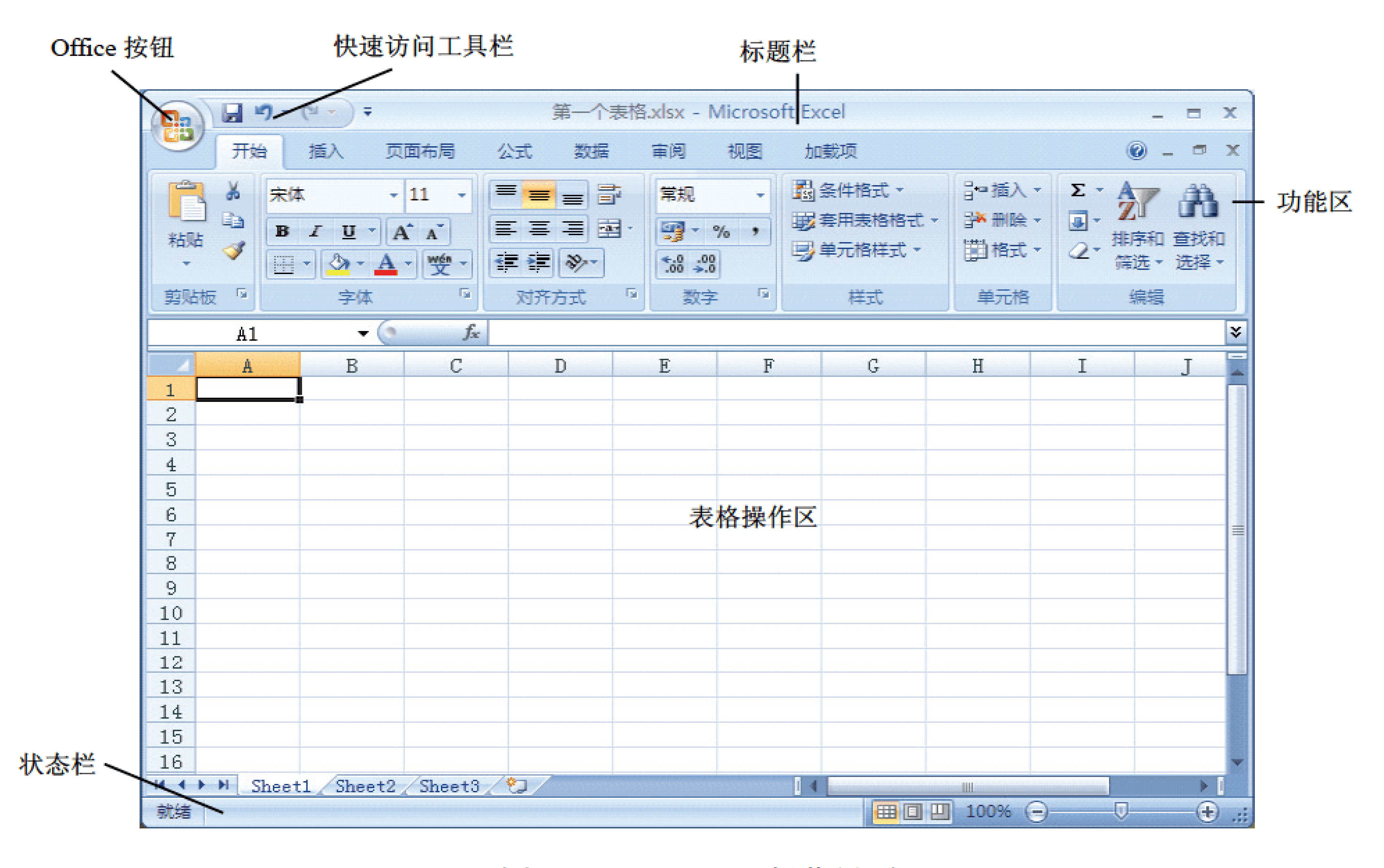


图 1-1　Excel 2007 操作界面

与以前版本相比，Excel 2007 的界面有了非常大的改变。Excel 2007 采用全新的外观，新的用户界面用简单明了的单一机制取代了 Excel 早期版本中的菜单、工具栏和大部分任务窗格。

使用过以前版本的 Excel 读者可能不太习惯 Excel 2007 的操作界面，因为在常见的 Windows 系统中，不管是 Office 应用程序，还是其他应用软件，基本上都是菜单和工具栏的形式。但进一步熟悉和使用 Excel 2007 后，才能体会到这种命令组织的独特和方便之处，这种方式将众多的命令巧妙地组合在一起，并使常用的命令一目了然地出现在界面中。下面简单介绍 Excel 2007 界面中各组成部件的使用。

1．Office按钮

2007 版的 Microsoft Office System 程序的用户界面已经全面重新设计。【Office 按钮】取代了原有版本中的【文件】菜单，它位于 Excel 2007 界面的左上角，如图 1-2 所示。

单击【Office 按钮】时，将看到与 Excel 早期版本相同的【打开】、【保存】和【打印】等基本命令，还新增了【发布】等菜单命令，另外在【Office 按钮】的下拉菜单右侧，还显示出了最近使用的文档。【Office 按钮】的下拉菜单如图 1-3 所示。

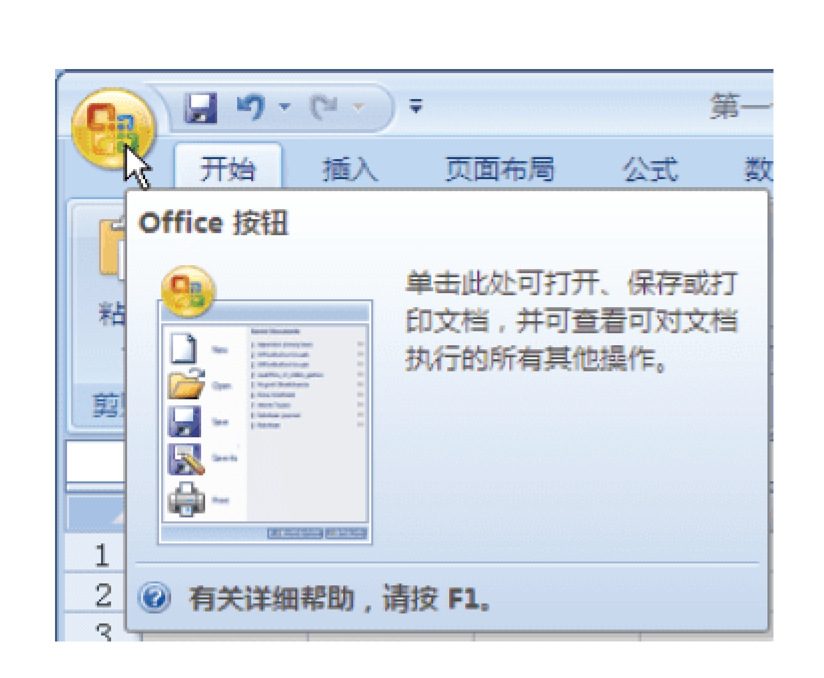


图 1- 2 【Office 按钮】

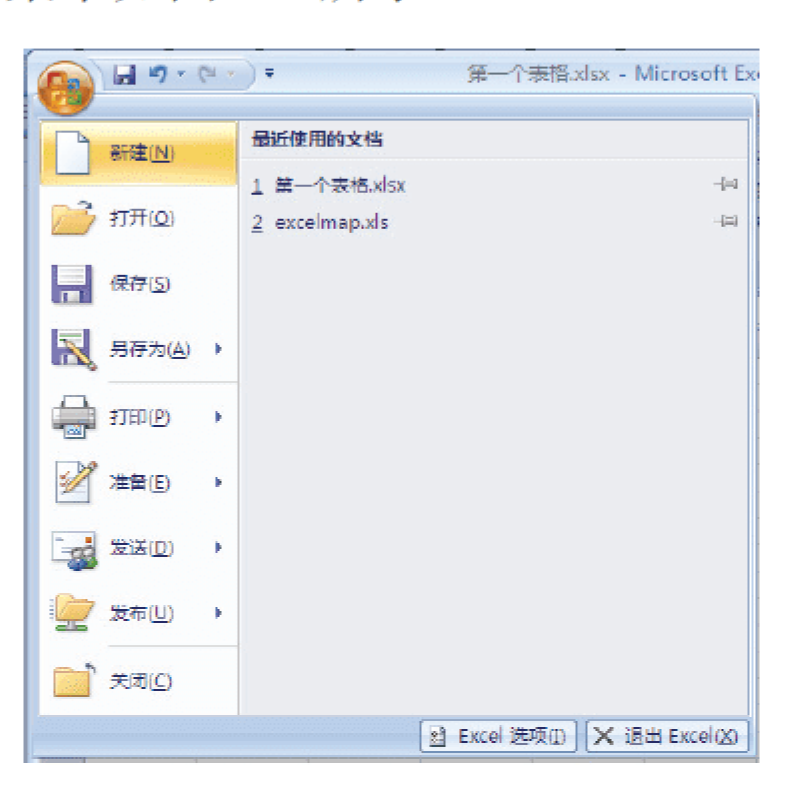


图 1-3 【Office 按钮】的下拉菜单

2．标题栏

Excel 2007 的标题栏与以前版本也不同。默认状态下，标题栏左侧显示“快速访问工具栏”，在标题栏中间显示当前编辑表格的文件名称。启动 Excel 时，将创建一个空白的工作簿文件，默认的文件名为 Book1。若按下快捷键 Ctrl+N 将继续创建新的工作簿文件，Excel 将分别以 Book2、Book3 等作为新文件的文件名。

3．快速访问工具栏

该工具栏可能是 Excel 2007 中与以前版本的工具栏最相似的部分了。在该工具栏中可以放置常用的命令按钮，以方便快速调用对应的功能。

通常，“快速访问工具栏”在 Excel 工作簿界面的左上方，也可以在“Excel 选项”对话框的【自定义】页面中选中【在功能区下方显示快速访问工具栏】复选框，将其放置在功能区下方。右击任何命令按钮，从弹出的快捷菜单中选择【添加到快速访问工具栏】命

令，将该命令按钮添加到其中，也可右击【快速访问工具栏】命令，在弹出的快捷菜单中选择【从快速访问工具栏中删除】命令，将该命令按钮从工具栏中删除。

4．功能区

在 Excel 2007 中，功能区是菜单和工具栏的主要替代控件。为了便于浏览，功能区包含多个围绕特定方案或对象进行组织的选项卡。每个选项卡上的控件进一步组织成多个组。功能区可比菜单和工具栏承载更加丰富的内容，包括按钮、图片库和对话框内容。

功能区的常规选项卡如图 1-4 所示。



图 1-4　功能区

- ❑ 选项卡：是面向任务设计的。
- ❑ 组：每个选项卡中的组都将一个任务分成多个子任务，如图 1-4 所示。
- ❑ 命令按钮：每个组中的命令按钮执行一个命令或显示一个命令菜单。

除了图 1-4 显示的选项卡外，还会在界面中看到对目前正在执行的任务类型有所帮助的另外两种选项卡。

- ❑ 上下文工具选项卡：使用户能够操作在页面上选择的对象，如表、图片或绘图。单击对象时，相关的上下文选项卡集以强调文字颜色出现在标准选项卡的旁边。如图 1-5 所示为选中图表时显示的选项卡。

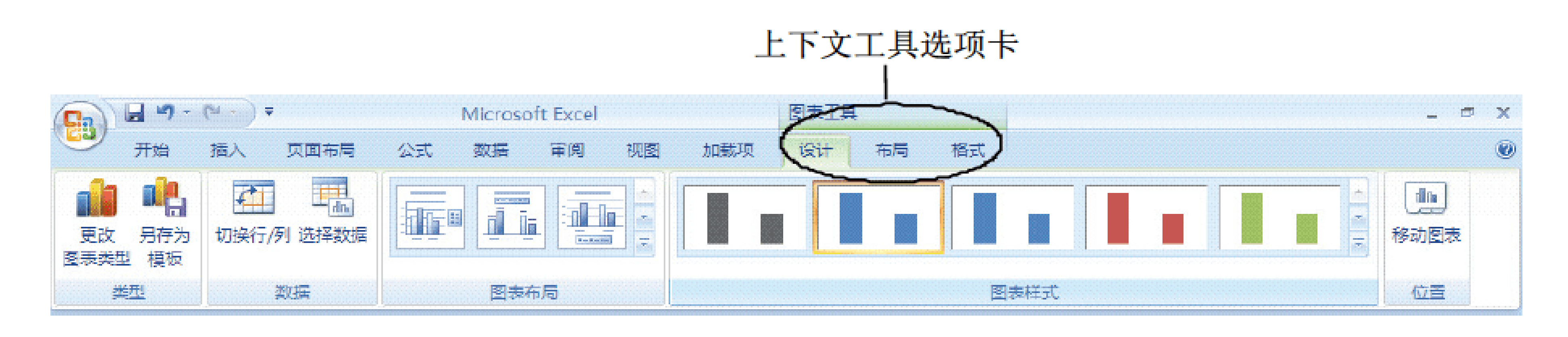


图 1-5　上下文工具选项卡

- ❑ 程序选项卡：当切换到某些创作模式或视图（包括打印预览）时，程序选项卡会替换标准选项卡集，如图 1-6 所示。



图 1-6　程序选项卡

5．对话框启动器

在功能区的某些组中，还显示有一个小的图标，如图 1-7 所示。这个图标称为对话框启动器。单击对

话框启动器将打开相关的对话框或任务窗格，其中提供与该组相关的更多选项。单击如图 1-7 所示的对话框启动器，并打开如图 1-8 所示的【设置单元格格式】对话框，在打开的对话框中，将根据对话框启动器所在组自动调出相应的选项组。

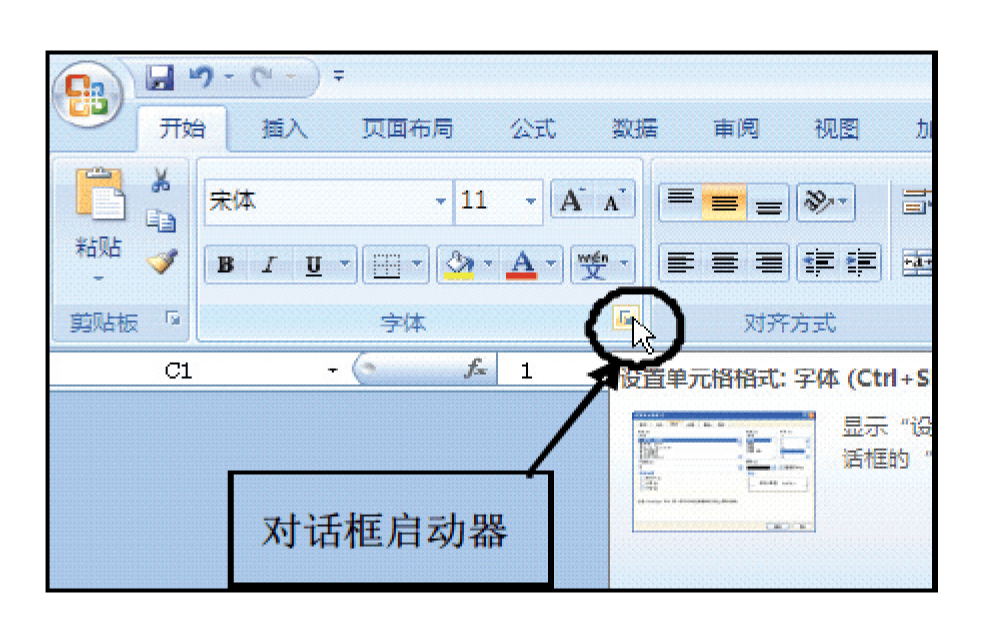


图 1-7　对话框启动器

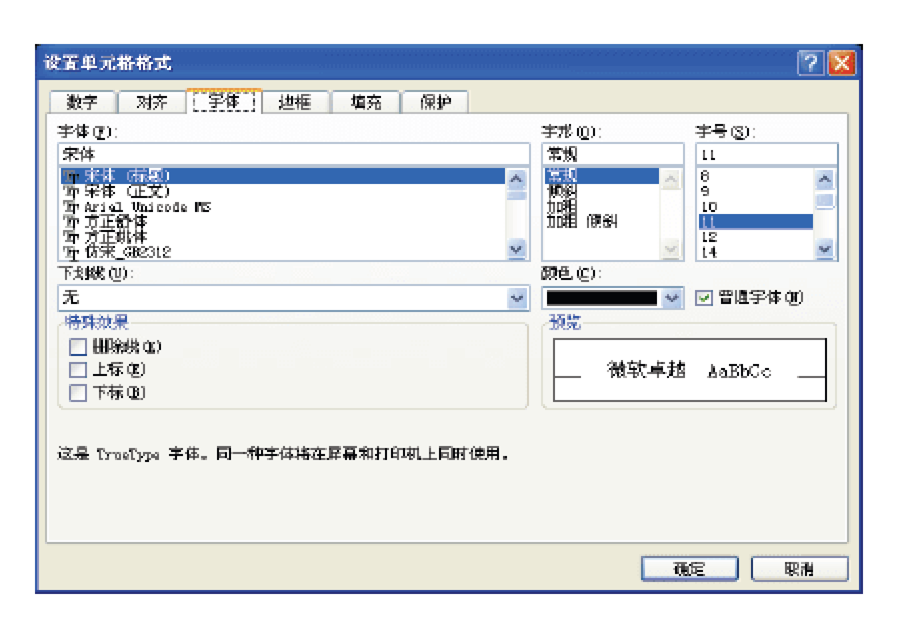


图 1-8　【设置单元格格式】对话框

6．状态栏

Excel 2007 的状态栏如图 1-9 所示。在状态栏中，可显示各种状态，也可进行很多快捷操作，例如，显示单元格中的统计数据、设置表格的视图方式、调整表格的显示比例、录制宏等。

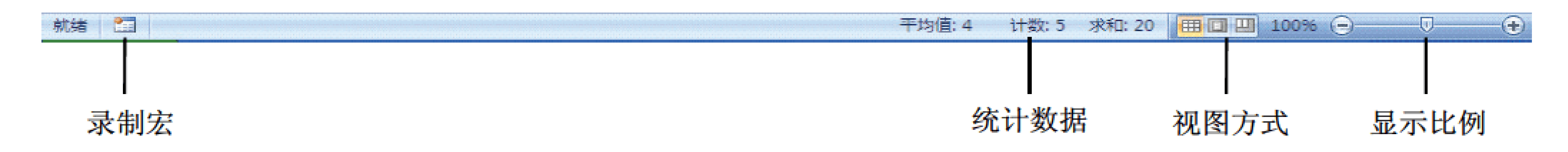


图 1-9　状态栏

右击状态栏，将弹出如图 1-10 所示的【自定义状态栏】菜单，单击对应的菜单项，可在状态栏中显示或隐藏对应的项目。



图 1-10　【自定义状态栏】菜单

1.1.4　使用功能区

当首次启动 Excel 2007 时，用户将会发现，Excel 以前版本的菜单栏和工具栏都不见了，取而代之的是“功能区”这个全新的用户交互界面。本节将详细介绍功能区的相关内容。

1．功能区的组成

功能区可以帮助用户快速找到完成某一任务所需的命令。命令被组织在逻辑组中，逻辑组集中在选项卡下。每个选项卡都与一种类型的活动（例如为页面编写内容或设计布局）相关。为了减少混乱，某些选项卡只在需要时才显示。例如，仅选择图片时，才显示【图片工具】选项卡。功能区有 3 个基本组成部分，如图 1-11 所示。



图 1-11　功能区

- 选项卡：默认情况下，功能区顶部有 8 个选项卡。每个选项卡代表在 Excel 中执行的一组核心任务。
- 组：每个选项卡都包含一些组，这些组将相关命令项显示在一起。
- 命令：命令是指按钮及用于输入信息的输入框或命令列表等。

2．智能显示命令

功能区上的命令是最常用的命令。Excel 2007 根据执行的操作显示一些可能用到的命令，而不是一直显示所有命令。例如，如果工作表中没有图表，则不会显示用于图表的命令。但在创建图表之后，就会出现【图表工具】标签，其中包含 3 个选项卡，分别是【设计】、【布局】和【格式】。在这些选项卡上，可看到处理图表所需的命令，如图 1-12 所示。这说明功能区对操作做出了响应。

使用【设计】选项卡可以更改图表类型或移动图表位置；使用【布局】选项卡可以更改图表标题或其他图表元素；使用【格式】选项卡可以添加填充颜色或更改线型。在完成图表后，单击图表区域外部，【图表工具】标签会消失。要想让它们重新显示，需单击图表内部，这些选项卡就会重新显示。

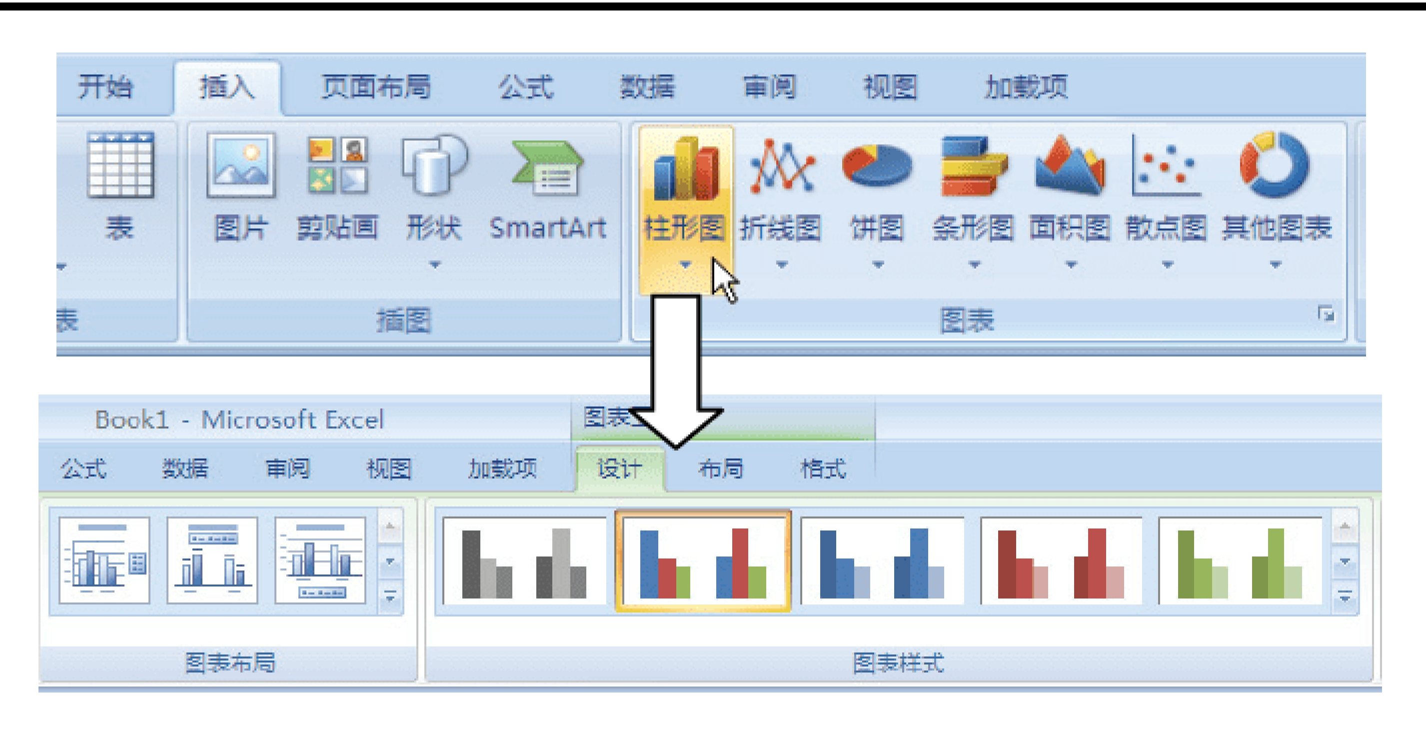


图 1-12　根据需要显示命令

3．最小化功能区

Excel 2007 的功能区是不能被删除的，也不能用早期版本的工具栏和菜单替换功能区。但是，可以最小化功能区以增大屏幕中可用的空间。最小化功能区有两种方式，一种是使功能区一直处于最小化状态，另一种是暂时最小化功能区。

- ❑ 始终使功能区最小化。

始终使功能区最小化的步骤如下：

（1）单击自定义快速访问工具栏右侧的按钮，弹出如图 1-13 所示的菜单。

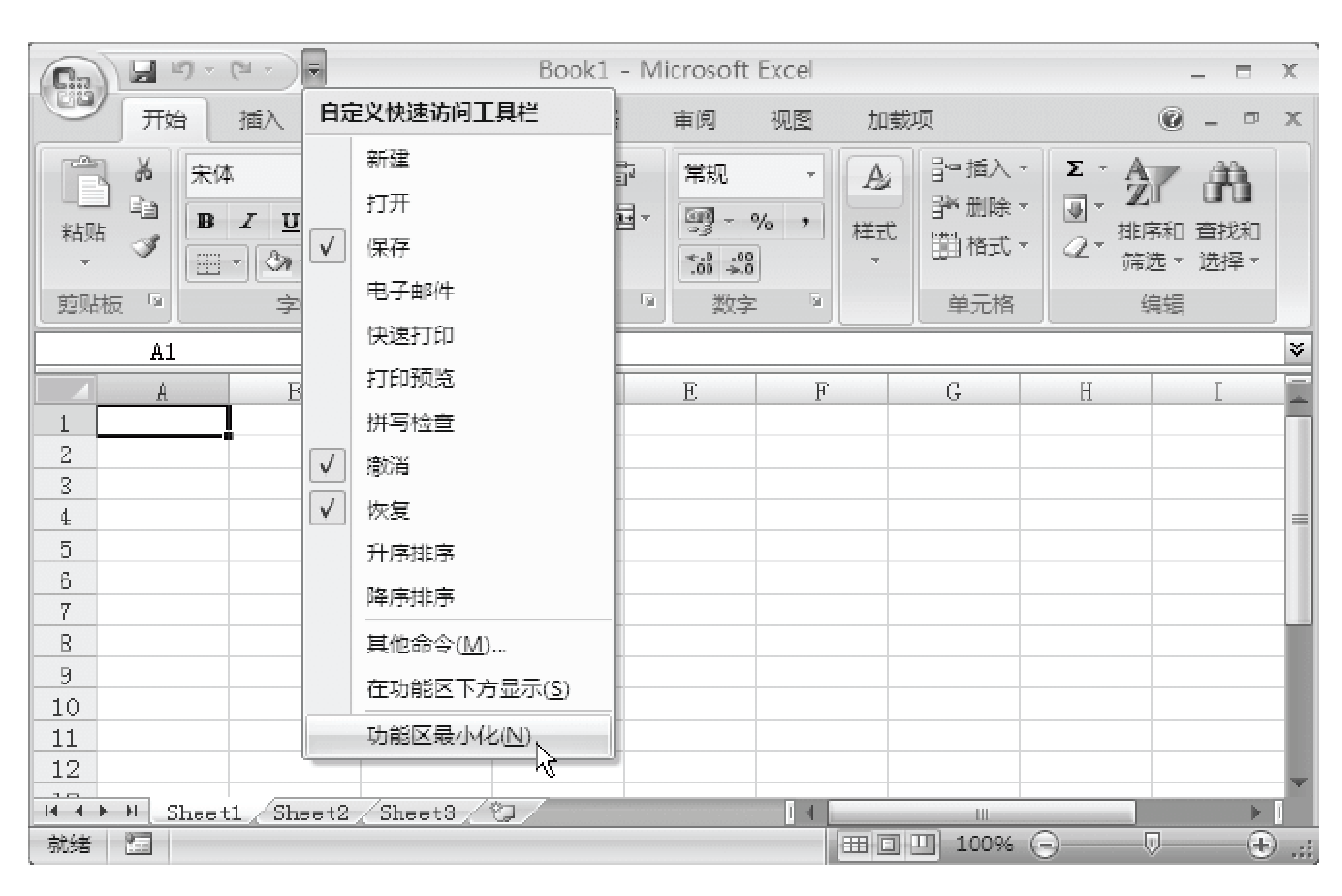


图 1-13　快速访问工具栏菜单

（2）在菜单列表中，选择【功能区最小化】菜单命令。

技巧： 要在功能区最小化的情况下使用功能区，首先需单击要使用的选项卡，在弹出对应的功能区内单击要使用的选项或命令即可。使用完对应的命令后，功能区将返回到最小化状态。

- ❑ 暂时最小化功能区。

当需要暂时将功能区最小化时，直接双击活动选项卡的名称即可。再次双击此选项卡

可还原功能区。

🔔**技巧**：按 Ctrl+F1 组合键可快速最小化或还原功能区。

4．使用键盘操作功能区

Excel 2007 为功能区的每个命令都提供了访问键，通过按键盘上的几个键就可快速使用相关命令。一般只需单击 2～4 个键就可以访问到大多数命令。使用键盘操作功能区的步骤如下：

（1）按下并释放 Alt 键。在当前视图中每个可用功能的上方都显示访问键提示。

（2）按下要使用的功能上方的访问键上所显示的字母。

（3）根据所按下的字母，可显示其他键提示。

（4）继续按对应的访问键字母，直到按下了要使用的特定命令或选项所对应的字母为止。

🔔**注意**：在某些情况下，必须先按下包含该命令的组所对应的字母。

例如，使用键盘操作功能区来设置活动单元格的字体为楷体。具体操作步骤如下：

（1）在 Excel 中按下并释放 Alt 键，功能区上方各选项卡中将显示出对应的访问键提示，如图 1-14 所示。

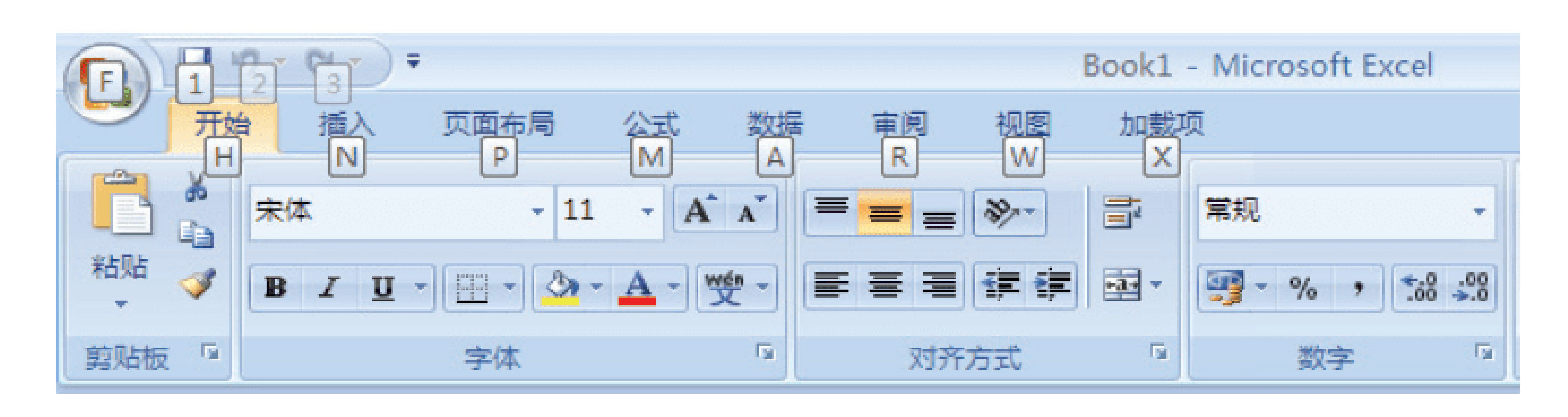


图 1-14　显示访问键提示

（2）使用开始选项卡中的命令，按下键盘上的 H 键，则【开始】选项卡中各命令的访问键将显示出来，如图 1-15 所示。

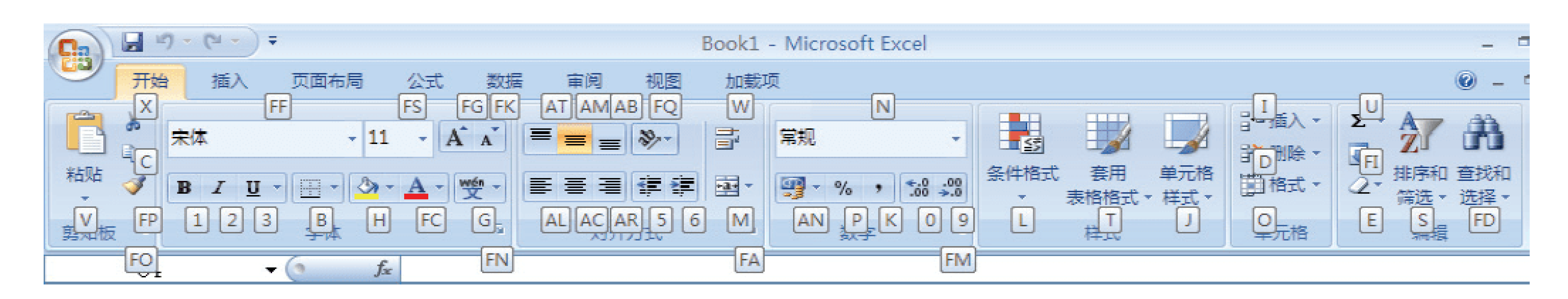


图 1-15　【开始】选项卡的访问键提示

（3）要选中字体下拉框，按下键盘上的 FF 键（按两次 F 键），则字体下拉框被选中，再按下键盘中的下方向光标键↓，找到字体为楷体后按 Enter 键，即可将活动单元格的字体设置为楷体，如图 1-16 所示。

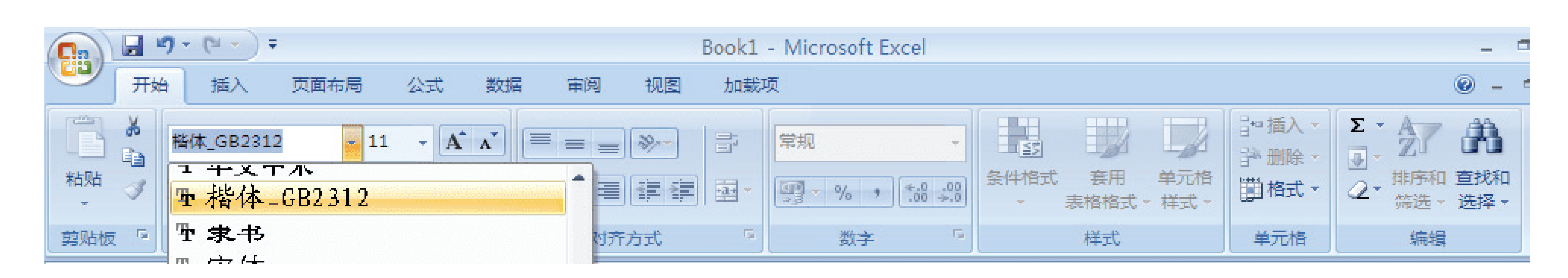


图 1-16　选择字体

技巧：为了和 Excel 以前版本兼容，Excel 以前版本的快捷键仍然可用。

5. 不同分辨率下的功能区

在 Excel 2007 中，根据屏幕分辨率不同，功能区的显示状态也不一样，具体情况如下。

如果屏幕设置为低分辨率，例如 800×600 像素，则功能区上会有一些组会改变命令的排列方式，有一些组仅显示组名称，而不显示该组中的命令。这时需要单击组按钮上的箭头才能显示命令。

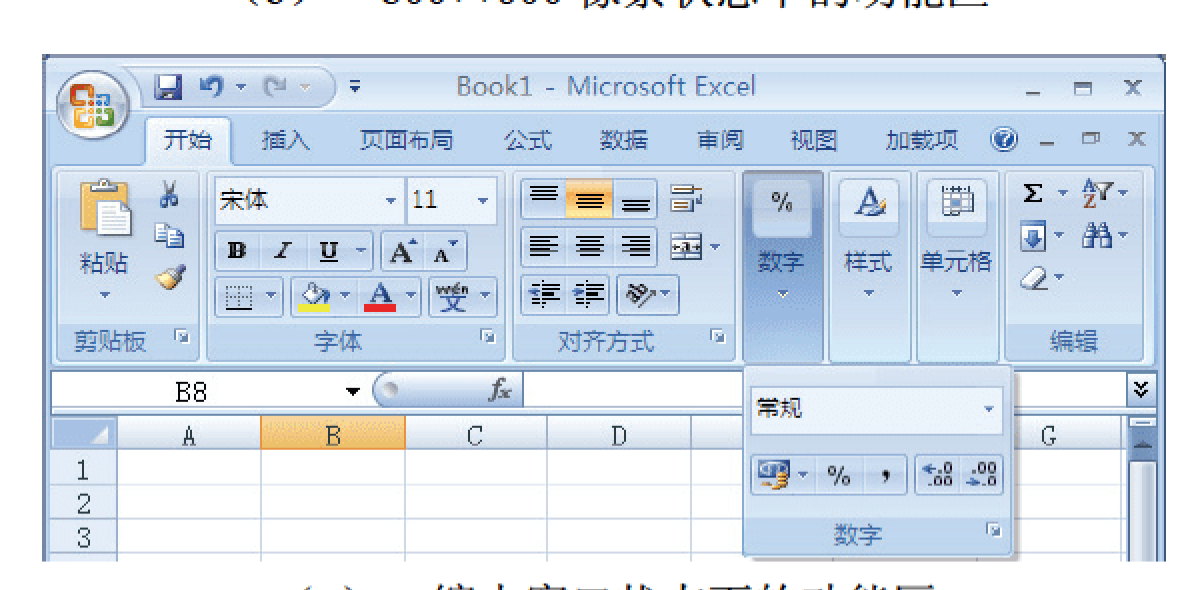
例如，在【开始】选项卡上，【数字】组中有几个命令。在较高分辨率下，会看到【数字】组中的所有命令排为两行，如图 1-17（a）所示。而在 800×600 的分辨率下，将会看到【数字】组中的命令按钮排列成三行了，如图 1-17（b）所示。将 Excel 窗口缩小后的功能区如图 1-17（c）所示，这时【数字】组中的命令不见了，单击【数字】组按钮上的箭头，将在下方显示出该组的命令。



（a）　1024×768 像素状态下的功能区



（b）　800×600 像素状态下的功能区



（c）　缩小窗口状态下的功能区

图 1-17　不同分辨率下的功能区

1.2　用 Excel 开发应用程序的优势

微软公司的 Excel 电子表格现在已成为最流行的软件包之一。大量的商务人员都在使用 Excel 来管理他们的商务数据，因此，大多数人员都接受过 Excel 的使用培训。

大多数用户使用 Excel 时，仅仅是将数据键入到工作表的单元格中，然后经过计算把结果显示在不同的单元格或图表中。工作表是一个数据输入和输出的用户界面，通过它来完成一些日常工作非常容易。其实，Excel 内部为数据的录入提供了极其丰富和完备的功能，其中包括对单元格中数据的编辑、有效性检查和格式的设置等。同时，利用图表和单元格格式的设置以及各种绘图工具可以很好地控制数据的输出形式。

Excel 不仅仅是一种电子表格，程序设计人员还可对其进行二次开发。微软公司在 Excel 中引入了 VBA（Visual Basic for Applications，通常称为 VBA，是一种应用程序自动化语言），使 Excel 成为一个引人注目的开发平台。用 Excel 开发的应用程序与用 VB（Visual Basic，简称 VB）、C++、Java、.NET 等语言开发的应用程序一样，成了许多公司重要商业软件的核心组成部分。

使用 Excel 开发应用程序具有以下几方面的优势。

- 节省用户培训费用：Excel 作为最基本的一种办公自动化软件，普及程度高，用户对其操作界面和操作方式都很熟悉，基于 Excel 平台开发的系统可让用户快速上手。
- 加快开发速度：在其他程序语言开发环境中，要设计类似于 Excel 的窗体界面，所需要的代码是难以想象的，而开发基于 Excel 的应用程序则要简单得多。
- 方便地控制 Excel：Excel 提供了完善的对象模型。几乎每种在 Excel 用户界面中能够完成的功能，都能通过使用 Excel 对象模型中的对象进行编程实现。
- 提高开发效率：Excel 提供了应用系统基本模块，例如，文件处理、打印和文本编辑等功能，用其他程序设计语言开发应用程序时，这些基本模块的开发就要占用很多的时间，所以使用 Excel 开发应用程序能提高开发的效率。
- 简化应用程序：Excel 内置大量函数，直接调用 Excel 的函数可简化程序，提高效率。
- 方便地处理大量数据：Excel 作为电子表格软件，其本身就可以处理大量的数据，对于数据量非常大的系统，Excel 还可连接到各种数据库中获取数据，然后在表格中进行分析处理。
- 快速创建动态的分析图表：使用 Excel 可方便、快速地创建图表，对工作表中的数据进行分析。使用 VBA 代码可创建、控制这些图表。

1.3　Excel 应用程序结构

使用 Excel 可开发各种应用程序，例如，简易财务系统、工资管理、固定资产管理、

工程预算、客户管理等 OA 系统。在 Excel 中使用 VBA 创建这些应用程序时，一般没有一个固定的结构，但大多数情况下，都有一些通用的构成模块，本节将简单介绍常见的 Excel 应用程序的构成。

1.3.1 Excel 应用程序的构成

一般情况下，二级开发的 Excel 应用程序都是以 Excel 工作簿的形式发布的。从用户角度看，与打开其他 Excel 工作簿的操作类似，不同之处在于，二次开发的 Excel 应用程序具有更多的智能性，能提高用户的操作效率。

而从开发者角度看，二次开发的 Excel 应用程序一般由用户窗体、工作表、模块和类模块等部分构成。

- 用户窗体：在 Excel VBA 应用程序中，用户窗体作为最常用的用户界面被大量使用。使用用户窗体可将用户与工作表中的数据进行隔离，防止数据被意外修改并隐藏工作表中的敏感数据，使限制权限的用户只看到应该操作的数据。一般系统中，常见的用户窗体有登录窗体、数据录入窗体、图表显示窗体等。
- 工作表：工作表是 Excel 用户最熟悉的工作界面，用于保存和显示程序的数据，是程序的主体部分。在开发 Excel 应用程序时，一般先在工作表中制作出表格的格式，并设置好样式，再通过 VBA 代码获取表格中的数据，经过加工处理后将其填入相应的单元格，供用户进行查看、打印输出等操作。
- 模块：模块是保存 VBA 代码的地方，可保存程序的通用过程，供其他过程调用。例如，录制宏的代码就保存在模块中，开发人员大部分时间都在模块中编写 VBA 代码。
- 类模块：类模块用来保存自定义对象的 VBA 代码。在 Excel VBA 中，除了可以使用系统提供的对象外，还可以通过自定义类来创建自定义的对象，自定义的类必须保存在类模块中。大多数应用程序都不使用类模块。

1.3.2 面向对象编程机制

VB 是面向对象的编程语言，而 Excel 中的 VBA 是 VB 的一个子集，也支持面向对象的编程机制。

在面向对象的编程机制中，程序中的每个部件都是一个对象。如窗体、按钮、工作表、单元格等都是对象，开发人员通过编写代码操作这些对象，即可完成对 Excel 的控制。

在 Excel VBA 中，通过事件驱动提供开发人员与系统之间的接口，开发人员通过编写事件过程来处理产生该事件时希望系统完成的工作（例如，单击鼠标就打开一个窗口）。

在传统的或过程化的应用程序中，应用程序自身控制了执行哪一部分代码和按何种顺序执行代码。从第一行代码执行程序并按应用程序中预定的路径执行，必要时调用过程。在事件驱动的应用程序中，代码不是按照预定的路径执行，而是在响应不同的事件时执行不同的代码片段。事件可以由用户操作触发，也可以由操作系统或其他应用程序的消息触发，甚至由应用程序本身的消息触发。这些事件的顺序决定了代码执行的顺序，因此，应

用程序每次运行时所经过的代码路径都是不同的。

因为事件的顺序是无法预测的，所以在代码中必须对执行时的各种状态作一定的假设。当作出某些假设时（例如，假设在运行时处理某一输入字段的过程之前，该输入字段必须包含确定的值），应该组织好应用程序的结构，以确保该假设始终有效（例如，在输入字段中有值之前，禁止使用启动该处理过程的命令按钮）。

在 Excel 中使用 VBA 开发应用程序，实质就是编写程序中各对象不同事件的代码。事件是对象识别的动作，例如，打开 Excel 工作簿、切换当前工作表等都将产生相关事件。VBA 的对象有一个预定义的事件集，对每个事件都可编写一个事件处理过程。如果其中有一个事件发生，而且在关联的事件过程中存在代码，则 VBA 调用该代码。例如，在工作簿的 Open 事件中编写有代码，则打开 Excel 工作簿时将执行该事件中的代码。Excel 定义的事件很多，常见的事件有：

- 鼠标单击事件（Click）；
- 工作簿打开事件（Open）；
- 工作表激活事件（Activate）；
- 单元格改变事件（Change）。

启动 Excel 应用程序后，工作簿、工作表或单元格等对象就准备接收事件。事件可由用户引发（例如键盘操作），可由系统引发（例如定时器事件），也可由代码间接引发（例如当代码激活其他工作表产生的事件）。

1.4　Excel 应用程序开发流程

与使用其他程序设计语言开发应用程序相同，在进行 Excel 应用程序二次开发时，也可以使用成熟的程序开发方法，以提高其开发效率。开发人员必须掌握正确的开发手段，了解软件开发的主要过程，这样对软件项目才能有清醒的认识，才能达到事半功倍的效果。本节就 Excel 应用程序开发过程中的一些方法进行简单的介绍。

1.4.1　开发前的准备工作

在进行 Excel 应用程序开发时，首先要编写系统任务书，主要规定应用程序的开发目标、主要任务、功能、性能指标及开发人员和经费、进度等安排，以作为系统设计开发和检验的基本依据。

针对具体情况，对应用程序的细节进行具体分析，必要时还要进行实地调研，与客户进行沟通，然后编写出需求分析文稿。

需求分析的任务不是确定应用程序怎样做的问题，而是确定需要完成哪些工作的问题。需求分析阶段的主要任务包括以下几个方面。

- 功能需求：给出应用程序必须完成的所有功能。
- 环境需求：用户的计算机硬件环境、软件环境和 Excel 的版本等。
- 界面需求：应用程序的用户界面是直接面对用户的，所以界面设计是用户能否方

便、快捷地操作应用程序的关键之一。在需求分析阶段，应提出界面的需求。
- 安全保密需求：对客户信息的保密要求应在本阶段开始计划。
- 用户技术层次：在需求分析阶段，了解用户的技术层次，可为应用程序的开发提供一些辅助信息。

1.4.2 应用程序开发过程

有了系统任务书和需求分析报告后，开发人员就可以对 Excel 应用程序的实现进行系统分析，然后按照分析进行相应的程序设计、编写代码工作。

1. 系统设计

系统设计阶段是通过对用户需求进行调查分析，得出应用程序的功能、性能及数据要求，以确定 Excel 应用程序所需的工作表及表中的列数据、窗体等模块。

2. 设计用户交互界面

一个良好的应用程序必须有一个良好的界面。用户通过界面与应用程序进行交互。开发人员在设计界面时，一定要牢牢把握方便用户操作这一观点，并贯穿到设计界面中。

与以往版本不同的是，Excel 2007 中取消了菜单和工具栏（以往版本设计的工具栏和菜单将放在【加载项】选项卡的【自定义工具栏】和【自定义菜单栏】组中）。在 Excel 2007 中进行界面设计的方式主要有以下 3 种。
- 在工作表中添加控件；
- 用户窗体；
- 自定义功能区。

3. 代码设计

将用户界面设计好以后，接下来就需要编写界面中各部分的事件代码（如用户窗体中的按钮、功能区中的按钮等）了。

在 Excel 中设计 VBA 应用程序时，界面设计和代码设计一般是交替进行的，即设计好一个界面后就编写相应的代码。有时也可先录制修改好宏代码，再与工作表或用户窗体中的按钮进行绑定。本书后面的章节都是介绍界面设计和代码设计知识的，这是应用程序的核心部分。

1.4.3 系统测试

在创建了应用程序之后，必须对其进行测试，这是非常重要的一个步骤。测试和调试应用程序所花费的时间可能与开发系统的时间同样多。

对于一个开发完成的应用程序，在设计测试数据时，应尽可能多地考虑到各种不同的情况，不但要使用正常的合乎逻辑的数据测试应用程序的功能性，还应使用一些可能导致应用程序出错的数据测试应用程序的健壮性。

在设计测试数据的同时，应编写出测试数据的结果，并与应用程序进行实测时得到的数据进行对比，如果结果相同，则通过测试；否则，应检查并修改应用程序。

1.4.4　应用程序发布

通过测试后的应用程序就可发布给最终用户使用了。在发布时需要注意以下 3 个问题。

- Excel 版本：如果是在 Excel 2007 环境下开发的应用程序，并使用了 Excel 2007 的一些新功能（如自定义功能区），就需要用户使用 Excel 2007 版本。如果用户使用 Excel 2007 之前的版本，则需要将使用 Excel 2007 新功能部分的代码进行修改，并发布为以往的版本。
- 动态链接库：如果应用程序中使用了 ActiveX 控件，则需要考虑是否要将包含该 ActiveX 控件的 DLL 文件（或 OCX 文件）包含在应用程序中予以发布。
- 辅助文件：在一个大型的应用程序中，不可能只包括一个 Excel 工作簿文件，有时可能还需要使用其他辅助文件（如图片文件、数据库文件、帮助文件等），需要将这些文件包括在发布文件中，并且最好将其发布到其他盘符中进行测试，以检查在 VBA 代码中是否使用了绝对路径来引用相关的文件。

第 2 章　使 用 宏

创建和使用宏是 Excel 最强大的功能之一。宏是可用于自动执行任务的一项或一组操作。通过 VBA 编写的宏可控制 Excel 应用程序，对 Excel 的功能进行扩充。

要自动执行重复任务，可以在 Excel 中快速录制宏。也可以在 VBE（Visual Basic Editor，是编写 VBA 代码的工具）中编写自己的宏脚本，或将所有或部分宏复制到新宏中来创建一个宏。在创建宏之后，用户可以将宏分配给对象（如工具栏按钮、图形或控件），以便能够通过单击该对象来运行宏。本章将介绍创建和管理宏的方法。

2.1　宏 简 介

在使用 Excel 的过程中，用户可能经常需要在 Excel 中进行重复的操作，并且这些重复任务将占用很多的时间。这时，有没有想过可能有更好的办法？如果经常需要执行一些重复操作，那么有必要了解一下有关宏的知识。

提示：用户在打开 Excel 工作簿时，可能看到过宏警告，因此，说到宏可能会联想到诸如病毒或编程等可怕的字眼。事实上，多数宏不仅无害，而且可以为用户节省大量的时间。宏的创建和使用非常简单。

2.1.1　什么是宏

宏是通过一次单击就可以应用的命令集。它们几乎可以自动完成用户在 Excel 中进行的各种操作。通过 VBA 代码对宏进行编辑修改，使宏还可以执行许多高级的、普通用户不能完成的任务。

宏是一种程序代码，即使用户不是开发人员也可以使用它们，甚至不需要知道任何编程知识。在 Excel 中可以创建的多数宏都是用 VBA 的语言编写的。VBA 宏就是本书所要详细介绍的内容。

2.1.2　使用宏的优点

宏可以节省时间，并可以扩展日常使用的程序的功能。使用宏可以自动执行重复的文档制作任务，简化繁冗的操作，还可以创建解决方案（例如，自动创建用户要定期使用的文档）。精通 VBA 编程的开发人员可以使用宏创建包括模板、对话框在内的自定义外接

程序，甚至可以存储信息以便重复使用。

例如，在一个具有几十个甚至上百个工作表的 Excel 工作簿中，要分别设置每个工作表的表头部分和数据部分的格式。如果在 Excel 中手工操作，假设每个工作表需要 1 分钟时间，则整个工作簿的格式设置也需要一两个小时才能完成，并且该项工作非常乏味。如果对其中的一个工作表设置格式，并将该操作录制为宏，然后编辑该宏，使之在整个工作簿中重复执行格式的设置，那么完成这项任务就不是几个小时了，只需几分钟就足够了。

2.1.3　创建宏的方法

在 Excel 中可使用两种方法来创建宏：一种方法是利用 Excel 操作环境中的宏录制器录制用户的操作；另一种方法是使用 VB 编辑器编写自己的宏代码。

- 利用宏录制器可记录用户在 Excel 中的操作动作，以便自动创建需要的宏。对于初学者，因为不熟悉 VBA 指令，使用该方法将非常方便。这也是初学者学习 VBA 指令的一种好方法。
- 使用 VB 编辑器可以打开已录制的宏，修改其中的命令，也可以在 VB 编辑器中直接输入命令创建宏。对于很多无法录制的命令（如创建新的窗体等），使用 VB 编辑器创建宏是唯一的方法。

在创建宏之后，可以将宏分配给对象（如按钮、图形、控件、快捷键等），这样执行宏就像单击按钮，或按快捷键一样简单。正是由于这种操作方便的特性，使用宏可以方便地扩展 Excel 的功能。如果不需要再使用宏，可以将其删除。

2.2　创 建 宏

可以通过录制宏和在 VBE 环境中编写代码这两种方式创建宏。Excel 2007 及 Excel 以前版本的宏使用相同的 VBA 代码。但是，在录制宏时，Excel 2007 与 Excel 以前版本有所不同。

2.2.1　在 Excel 2003 中录制宏

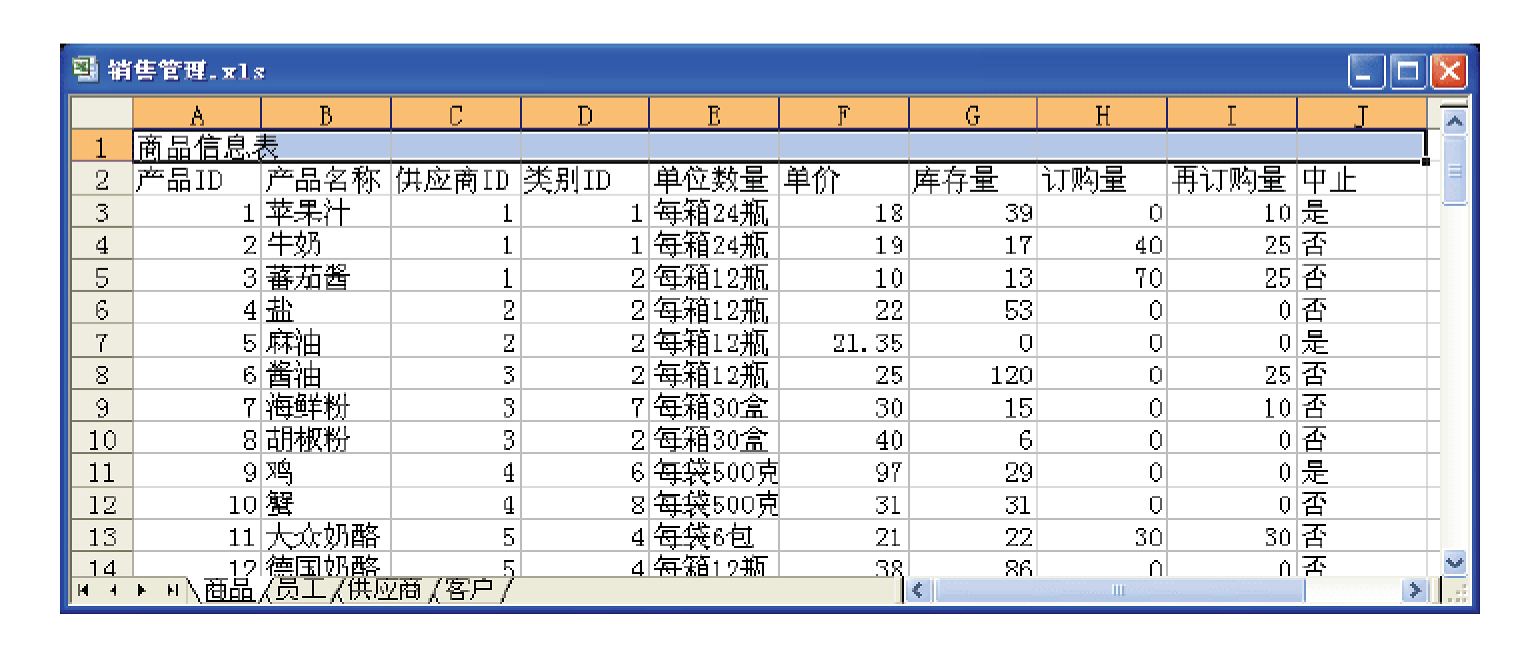
录制宏是创建宏的最简单、最常用的方法。宏录制类似于“记忆”用户在 Excel 环境中执行的操作，其方法与在盒式磁带上录制音乐类似。当按下录音键时，所有声音都存储在磁带上，直到按下停止键。录制宏的过程与此基本相同。按下【录制】按钮时，所执行的任务、使用的窗口和工具等都作为宏代码录制下来。

Excel 2003 及以前版本都使用菜单和工具栏方式执行相关操作。录制宏时通过单击主菜单【工具】|【宏】|【录制新宏】命令，即可将 Excel 中进行的操作用 VBA 代码记录下来。

技巧： 因 VBA 中的对象、属性非常多，对于初学者来说，确实不易记忆。通过录制宏，然后分析 Excel 自动记录的 VBA 代码，是学习 VBA 的一个有效方法。

下面以设置工作表表头格式为例，介绍录制宏的方法，具体操作步骤如下：

（1）启动 Excel 2003，打开工作簿“销售管理.xls”，并选中单元格区域“A1:J1”，如图 2-1 所示。



	A	B	C	D	E	F	G	H	I	J
1	商品信息表									
2	产品ID	产品名称	供应商ID	类别ID	单位数量	单价	库存量	订购量	再订购量	中止
3	1	苹果汁	1	1	每箱24瓶	18	39	0	10	是
4	2	牛奶	1	1	每箱24瓶	19	17	40	25	否
5	3	蕃茄酱	1	2	每箱12瓶	10	13	70	25	否
6	4	盐	2	2	每箱12瓶	22	53	0	0	否
7	5	麻油	2	2	每箱12瓶	21.35	0	0	0	是
8	6	酱油	3	2	每箱12瓶	25	120	0	25	否
9	7	海鲜粉	3	7	每箱30盒	30	15	0	10	否
10	8	胡椒粉	3	2	每箱30盒	40	6	0	0	否
11	9	鸡	4	6	每袋500克	97	29	0	0	是
12	10	蟹	4	8	每袋500克	31	31	0	0	否
13	11	大众奶酪	5	4	每袋6包	21	22	30	30	否
14	12	德国奶酪	5	4	每箱12瓶	38	86	0	0	否

图 2-1　选中单元格区域

（2）单击主菜单【工具】|【宏】|【录制新宏】命令，打开如图 2-2 所示【录制新宏】对话框。在【宏名】文本框中输入名称“设置表头格式”，并设置宏的快捷键为 Ctrl+t，然后选择好宏的保存位置。Excel 自动在【说明】文本框中填充备注信息（默认情况下将由 Excel 生成包括文稿作者、录制时间等信息），用户可在该文本框中输入宏的备注信息。

注意：宏的保存位置有 3 种：当前工作簿——宏只对当前工作簿有效；个人宏工作簿——宏对所有工作簿都无效；新工作簿——录制的宏保存在一个新建工作簿中，对该工作簿有效。

（3）单击【确定】按钮，将显示如图 2-3 所示的【停止录制】工具栏。






图 2-2　【录制新宏】对话框　　　　图 2-3　【停止录制】工具栏

（4）接下来可以在 Excel 中进行操作，设置工作表的表格格式。单击主菜单【格式】|【单元格】命令，打开【单元格格式】对话框。

（5）在该对话框中单击【对齐】选项卡，选中【水平对齐】列表框中【跨列居中】选项，如图 2-4 所示。

（6）在该对话框中单击【字体】选项卡，在【字形】列表框中选择“加粗”，在【字号】列表框中选中 20，如图 2-5 所示。

（7）单击【确定】按钮完成格式的设置。

（8）单击【停止录制】工具栏中的【停止录制】按钮，完成宏的录制。工作表设置表头格式后如图 2-6 所示。



图 2-4　设置跨列居中

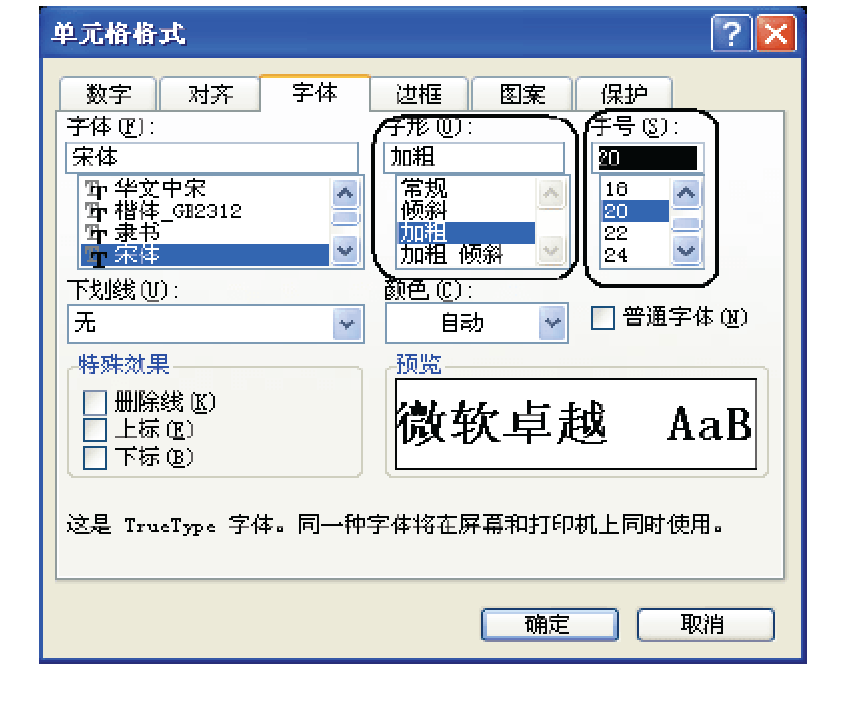
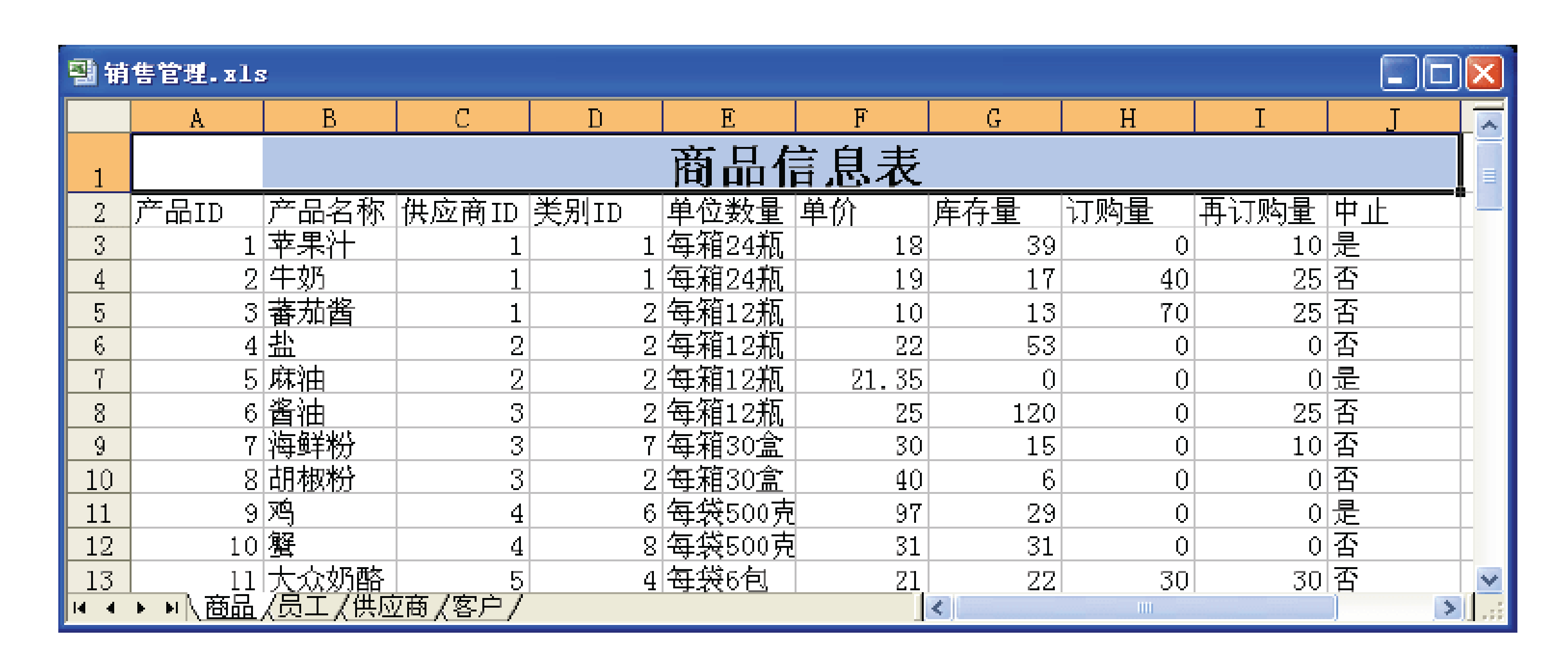


图 2-5　设置字体



	A	B	C	D	E	F	G	H	I	J
1	商品信息表									
2	产品ID	产品名称	供应商ID	类别ID	单位数量	单价	库存量	订购量	再订购量	中止
3	1	苹果汁	1	1	每箱24瓶	18	39	0	10	是
4	2	牛奶	1	1	每箱24瓶	19	17	40	25	否
5	3	蕃茄酱	1	2	每箱12瓶	10	13	70	25	否
6	4	盐	2	2	每箱12瓶	22	53	0	0	否
7	5	麻油	2	2	每箱12瓶	21.35	0	0	0	是
8	6	酱油	3	2	每箱12瓶	25	120	0	25	否
9	7	海鲜粉	3	7	每箱30盒	30	15	0	10	否
10	8	胡椒粉	3	2	每箱30盒	40	6	0	0	否
11	9	鸡	4	6	每袋500克	97	29	0	0	是
12	10	蟹	4	8	每袋500克	31	31	0	0	否
13	11	大众奶酪	5	4	每袋6包	21	22	30	30	否

图 2-6　设置表头后的格式

（9）按快捷键 Alt+F11 打开 VBE，可看到录制的宏代码，如图 2-7 所示。

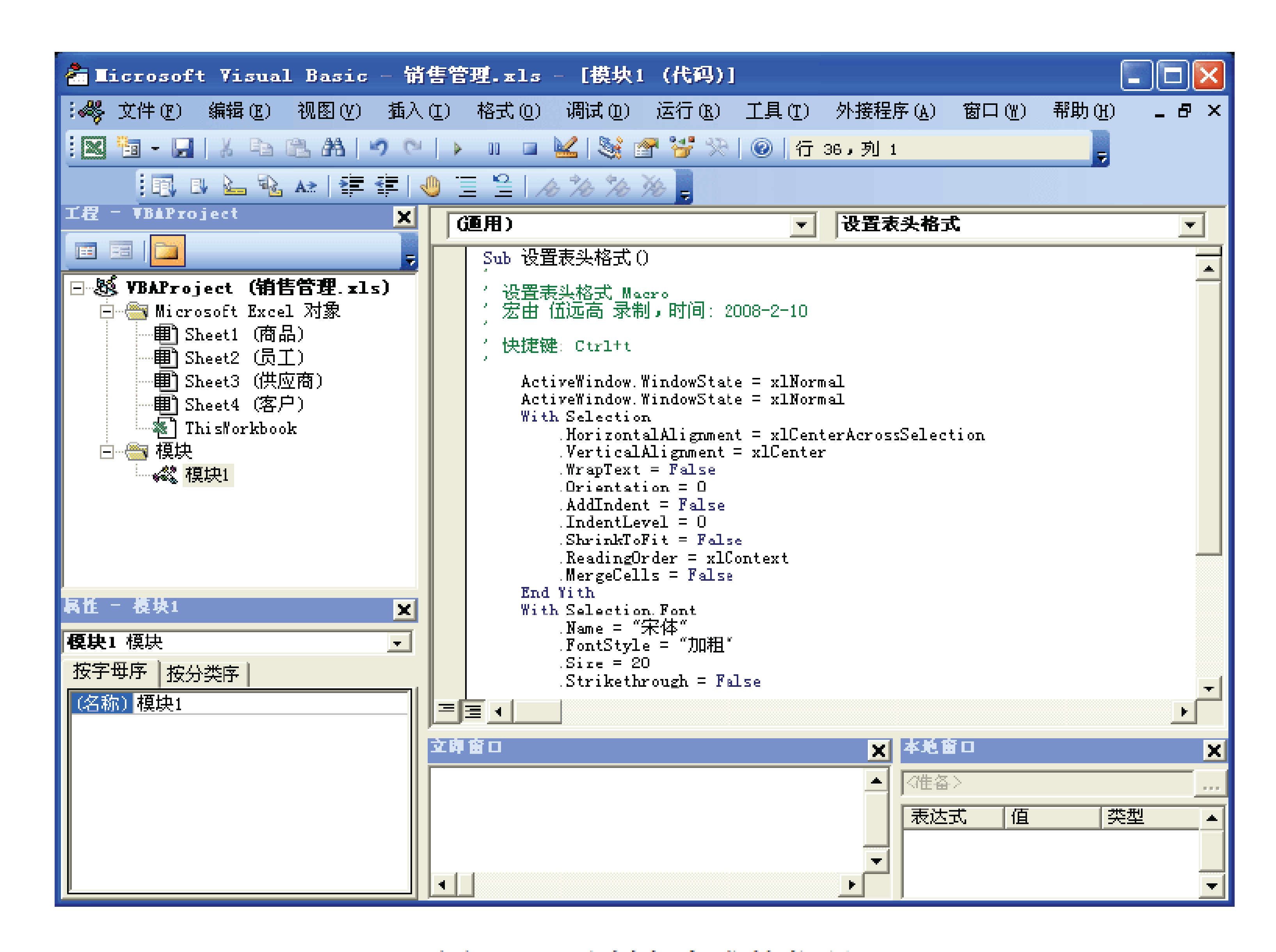


图 2-7　录制宏生成的代码

2.2.2 打开 Excel 2007 的录制宏功能

和以前版本的 Excel 相比，Excel 2007 采用了全新的面向结果的用户界面。以前版本中熟悉的菜单栏和工具栏消失了，被称为功能区（Ribbon）的面板取代。在功能区中，命令被组织在逻辑组中，逻辑组集中在选项卡下。每个选项卡都与一种类型的活动（例如为页面编辑内容或设计布局）相关。

要在 Excel 2007 中录制宏，需使用【开发工具】功能区中的相关命令。在 Excel 2007 的默认环境中，【开发工具】选项卡是隐藏的，如果要编写宏、运行以前录制的宏或创建与 Office 程序一起使用的应用程序，需要将【开发工具】选项卡显示出来。具体操作步骤如下：

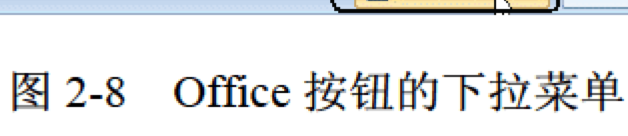
（1）启动 Excel 2007。

（2）在 Excel 操作界面上单击左上角的 Office 按钮，打开下拉菜单，如图 2-8 所示。

（3）单击下方的【Excel 选项】按钮，打开【Excel 选项】对话框，如图 2-9 所示。

（4）在该对话框中选择左侧的【常用】选项卡。在【使用 Excel 时采用的首选项】区域中选中【在功能区显示“开发工具”选项卡】复选框。




图 2-8 Office 按钮的下拉菜单

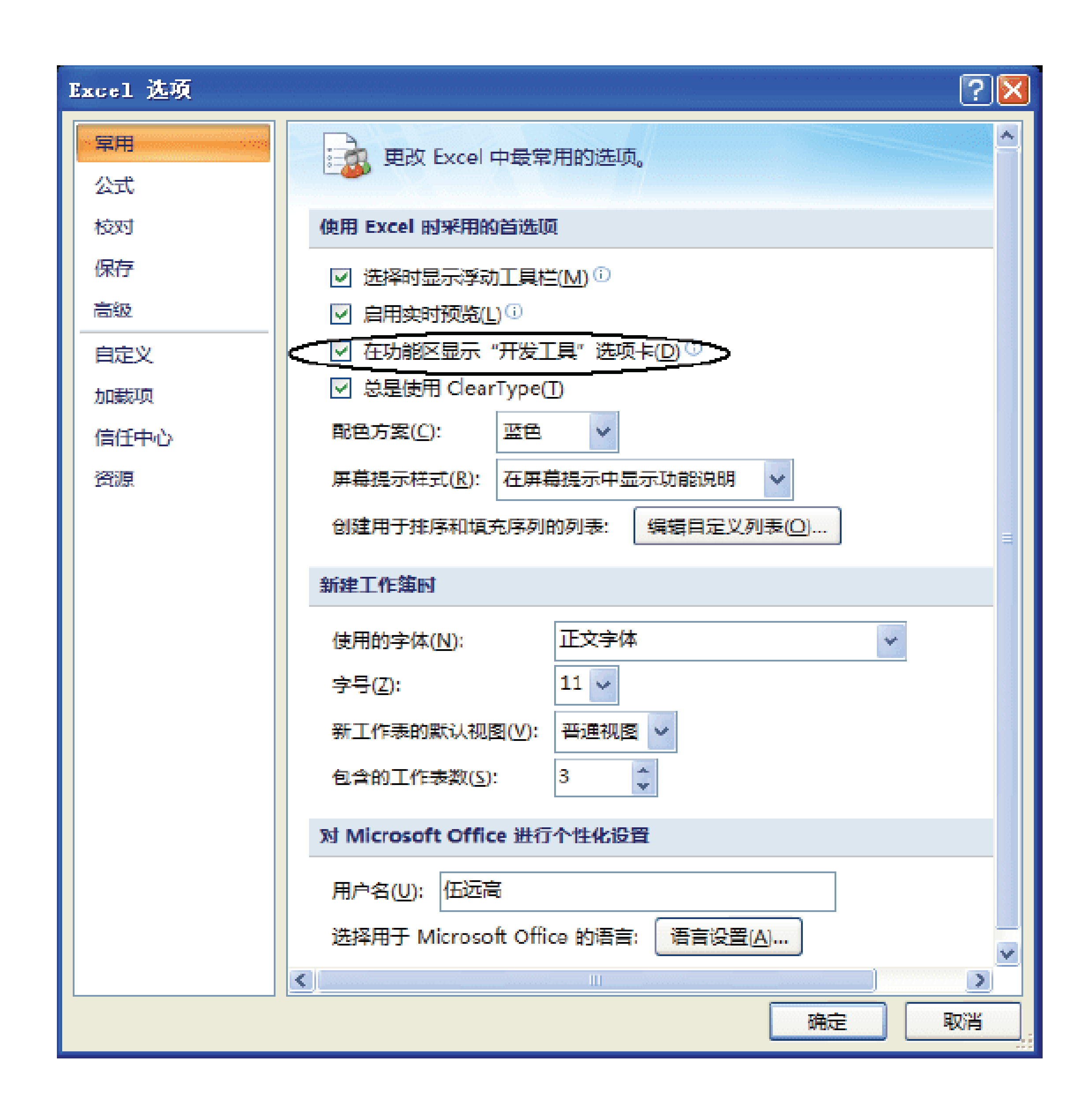


图 2-9 【Excel 选项】对话框

（5）单击【确定】按钮返回 Excel 2007 操作界面，可以看到【功能区】中新增加了一个【开发工具】选项卡，如图 2-10 所示。

（6）单击该选项卡，将显示如图 2-10 所示的【开发工具】功能区。在该选项卡的【代码】组中，有录制宏及对宏的设置等相关命令按钮。



图 2-10　【开发工具】选项卡

（7）在 Excel 2007 的状态栏中也提供了一个【录制宏】按钮，在默认状态下，该按钮未显示出来。要显示该按钮，右击状态栏将显示如图 2-11 所示的快捷菜单。

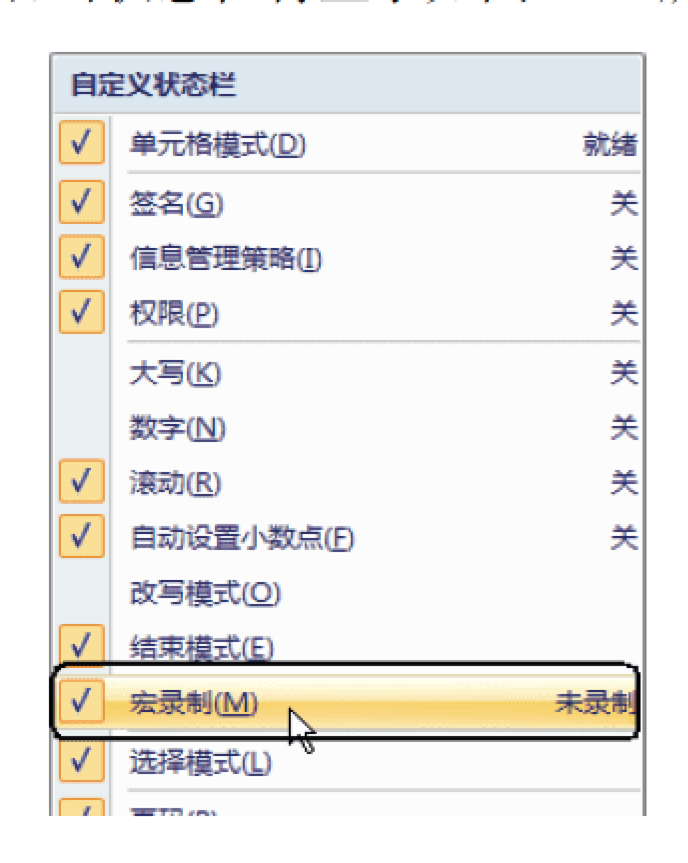


图 2-11　【自定义状态栏】快捷菜单

（8）单击【宏录制】命令，将在状态栏中显示如图 2-12 所示的【录制宏】按钮。在录制宏的过程中，该按钮将变为【停止录制宏】按钮。

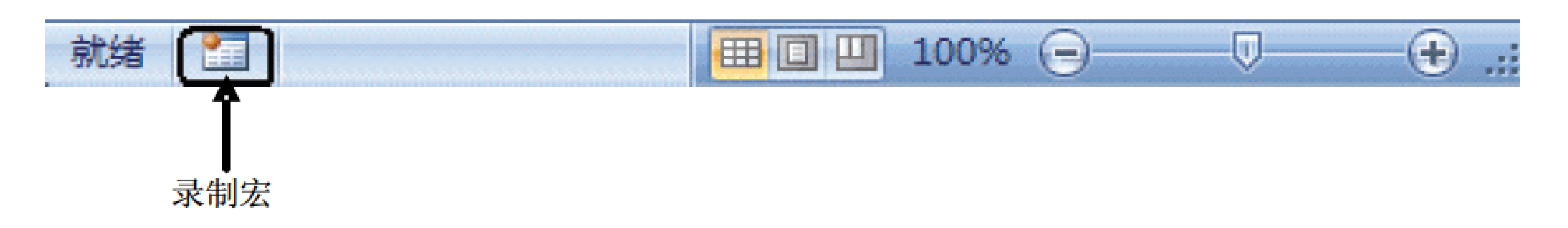


图 2-12　状态栏

2.2.3　在 Excel 2007 中录制宏

在 Excel 2007 中录制宏的操作与在 Excel 2003 中录制宏类似，下面演示在 Excel 2007 中录制宏，以了解两个版本中录制宏的相同和不同之处。

（1）启动 Excel 2007，打开如图 2-1 所示的工作簿。

（2）若 Excel 2007 的功能区中无【开发工具】选项卡，需要使用 2.2.2 节介绍的方法将【开发工具】选项卡显示出来。

（3）在工作表中选中单元格区域“A1:J1”。

（4）在【开发工具】选项卡的【代码】组中，单击【录制宏】按钮，如图 2-13 所示。

（5）打开【录制新宏】对话框，如图 2-14 所示。在【宏名】文本框中输入名称，并设

置好快捷键及保存位置等信息，单击【确定】按钮开始录制宏。此时状态栏中的【录制宏】按钮变为了【停止录制】按钮。

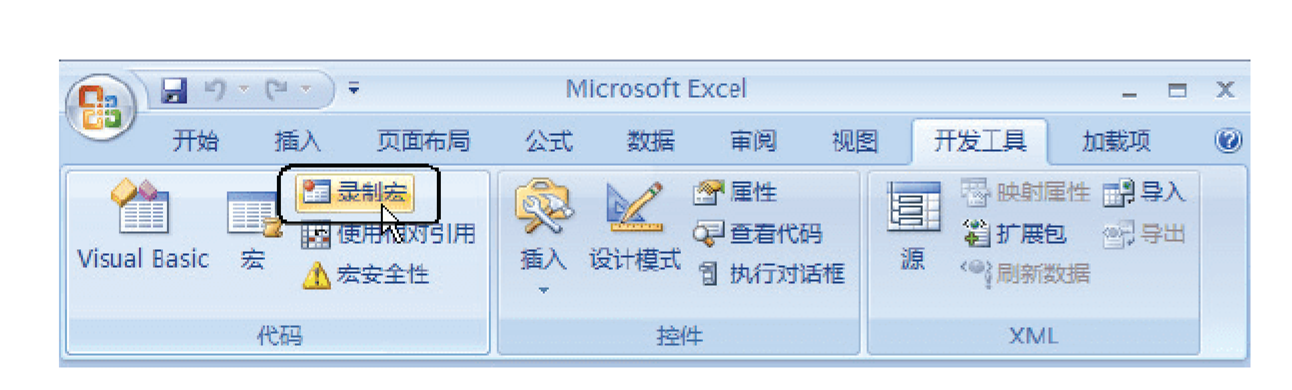


图 2-13 【录制宏】按钮

图 2-14 【录制新宏】对话框

（6）在功能区中单击【开始】选项卡，在该选项卡的【对齐方式】组中，单击【合并居中】按钮，将单元格区域“A1:J1”合并居中。

（7）在【开始】选项卡的【字体】组中，单击【加粗】按钮，在【字号】列表框中选中 20。

（8）在【开发工具】选项卡的【代码】组中，单击【停止录制】按钮，完成宏的录制操作。

提示：单击状态栏中的【停止录制】命令按钮，也可完成宏的录制。

（9）按快捷键 Alt+F11 打开 VBE，可看到录制的宏代码，如图 2-15 所示。

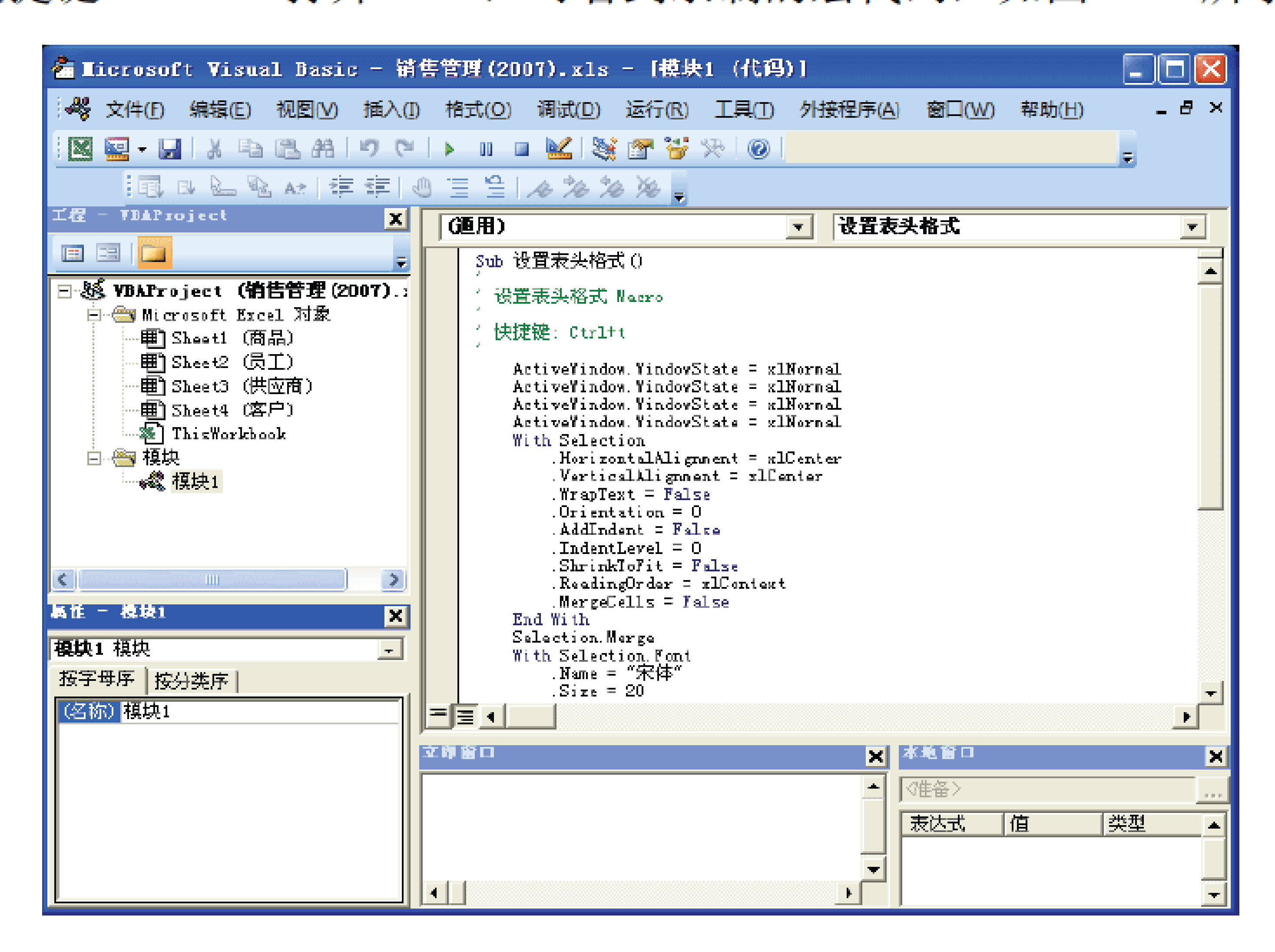


图 2-15 录制宏的代码

2.2.4 使用 VB 创建宏

使用宏录制器可在 Excel 中记录按顺序完成的操作。在实际使用时，经常需要在宏中

循环执行某一部分操作，使用宏录制器来创建这类宏是不可能的。这时就需要使用 VB 编辑器。在该编辑器中使用 VBA 代码可完成各种复杂的操作。

（1）启动 Excel 2007，如果软件已经启动，则新建一个工作簿。

（2）在【开发工具】选项卡的【代码】组中，单击 Visual Basic 按钮，打开如图 2-16 所示的 VB 编辑器窗口。

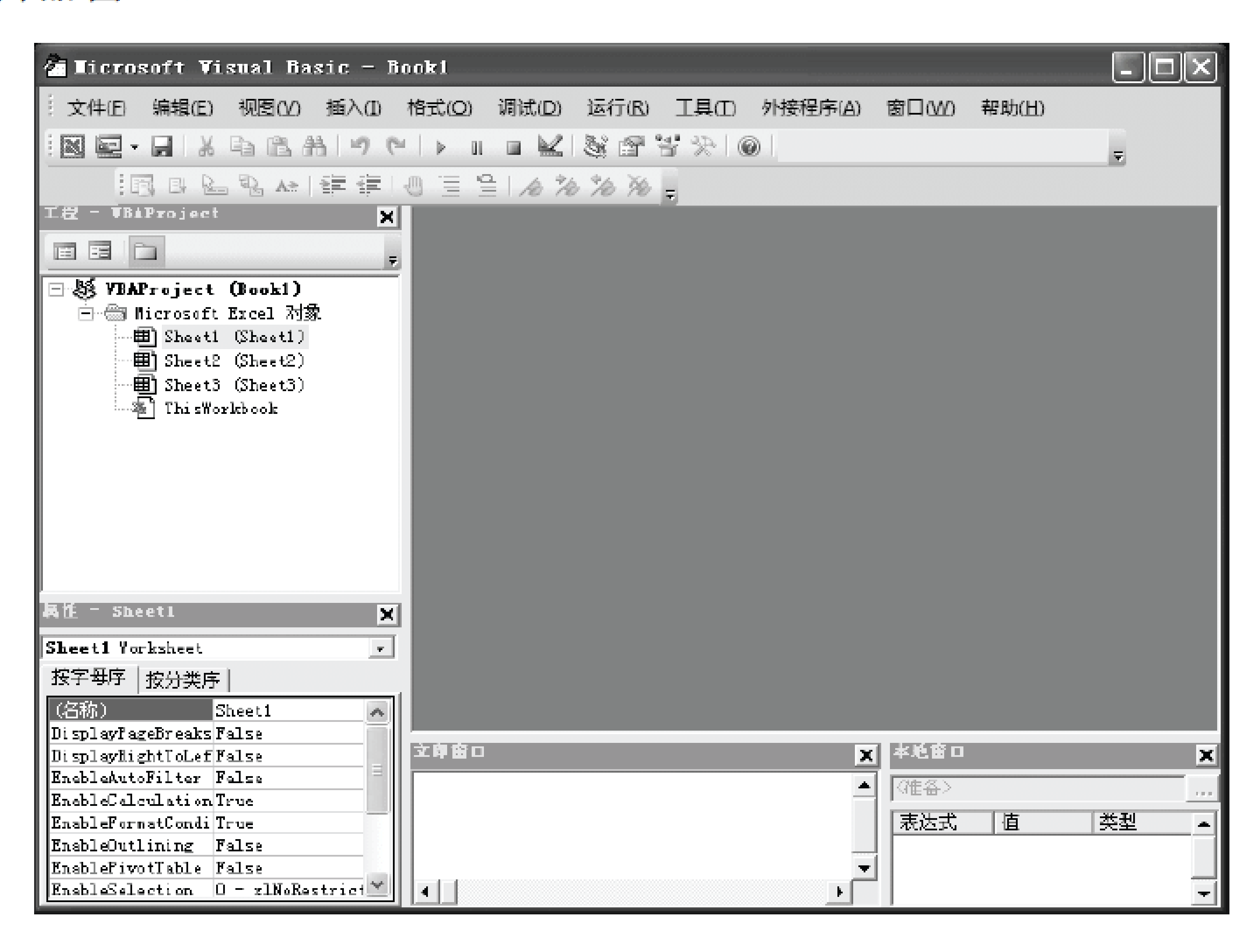


图 2-16　VB 编辑器

技巧：按键盘上的组合键 Alt+F11 也可打开 VB 编辑器。有关 VBE 编辑器的使用将在第 3 章中进行介绍。

（3）在 VBE 编辑器中，单击主菜单【插入】|【模块】命令，向工程中增加一个名为“模块 1”的模块。

（4）单击主菜单【插入】|【过程】命令，打开【添加过程】对话框，在【名称】文本框中输入相应内容，并在【类型】、【范围】选项区域中选择相应的选项，如图 2-17 所示。

（5）在【添加过程】对话框中单击【确定】按钮，将在【模块 1】代码窗口中添加一个名为“欢迎”的过程结构，如图 2-18 所示。

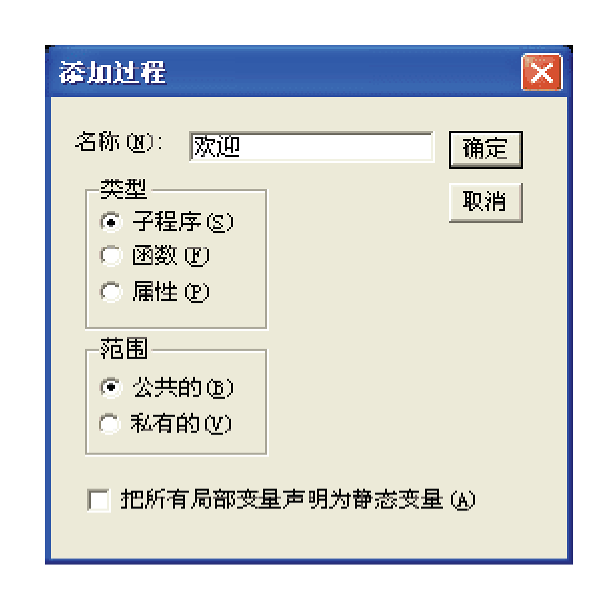


图 2-17　【添加过程】对话框

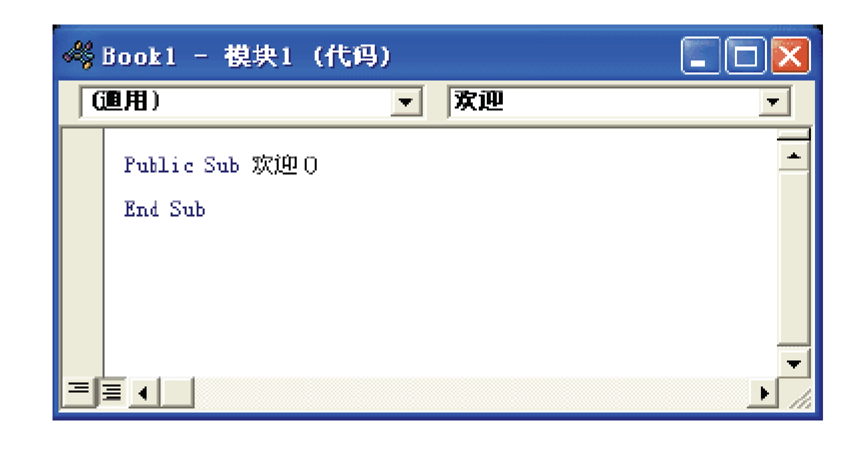


图 2-18　添加的过程结构

提示：有关过程的相关知识，参见本书第 2 部分中的介绍。

（6）在图 2-18 所示的过程结构中输入以下代码：

```
Public Sub 欢迎()
    Dim str1
    str1 = "欢迎您使用 Visual Basic! "
    MsgBox str1, vbOKOnly, "欢迎"
End Sub
```

（7）关闭 VB 编辑器，返回到 Excel 操作环境中。按快捷键 Ctrl+S 保存工作簿，完成宏的创建。

技巧：也可先通过宏录制器生成部分宏代码，再通过 VB 编辑器对生成的代码进行修改。

2.3 管理宏

在 Excel 2007 功能区【开发工具】选项卡的【代码】组中，单击【宏】按钮，打开如图 2-19 所示对话框，通过该对话框对 Excel 中的宏进行管理。

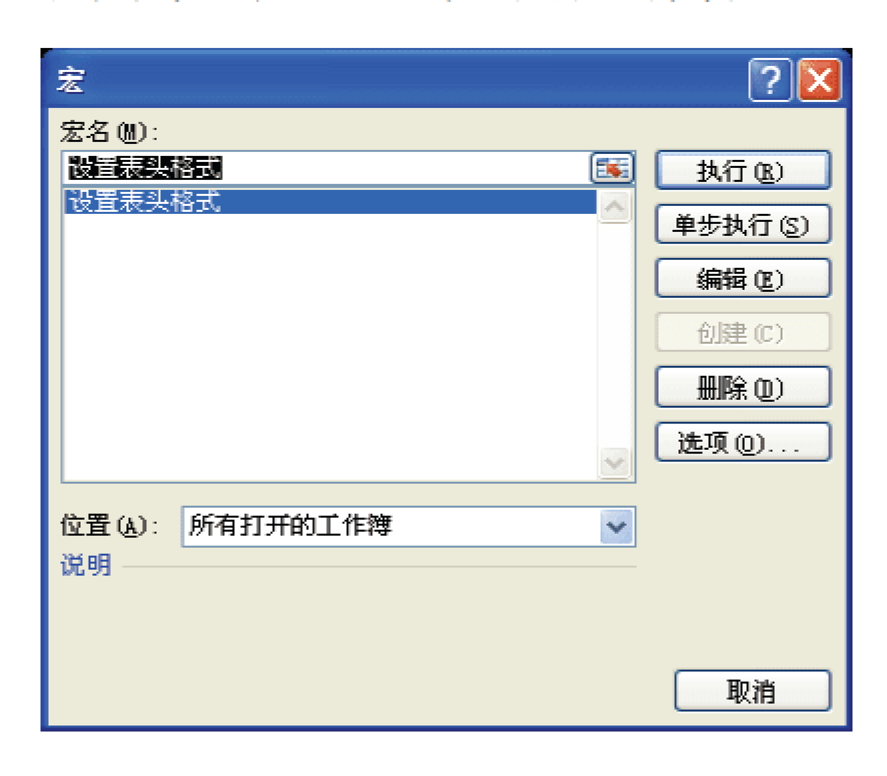


图 2-19 【宏】对话框

2.3.1 设置宏选项

用户在录制宏时，可为宏设置一个快捷键，在需要执行宏时可以快速调用。而用 VB 代码直接编写的宏则没有快捷方式，此时用户也可通过【宏选项】对话框设置其快捷键，具体操作步骤如下：

（1）在功能区【开发工具】选项卡的【代码】组中，单击【宏】按钮，打开如图 2-19 所示的对话框。

（2）在【宏名】列表框中选择需要设置选项的宏，单击对话框右侧的【选项】按钮，

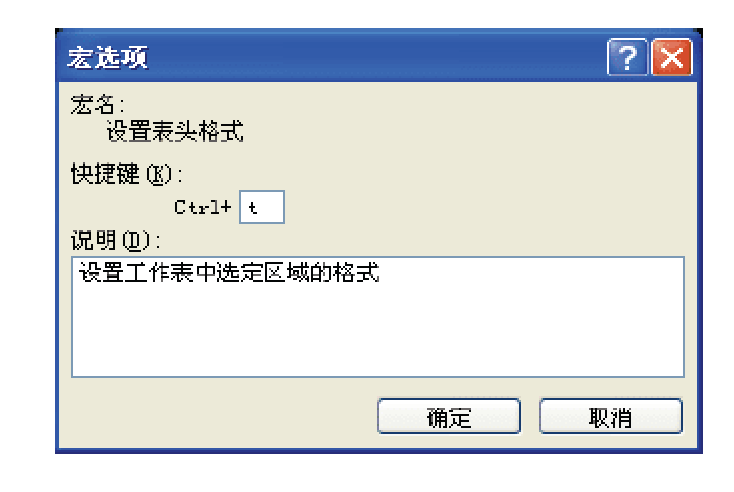


图 2-20　【宏选项】对话框

打开【宏选项】对话框，如图 2-20 所示。

（3）在【宏选项】对话框中可以设置快捷键和宏的说明，设置完成后单击【确定】按钮即可。

2.3.2　删除宏

对于工作簿中不需要的宏，也可将其删除。删除宏的步骤如下：

（1）在功能区【开发工具】选项卡的【代码】组中，单击【宏】按钮，打开如图 2-19 所示的对话框。

（2）在【宏名】列表框中选择需要设置选项的宏，单击对话框右侧的【删除】按钮，将打开如图 2-21 所示的提示信息，单击【是】按钮完成宏的删除，单击【否】按钮不删除宏。

图 2-21　提示信息

提示：在 Excel 中按快捷键 Alt+F11 进入 VBE，选中需要删除的宏代码，按 Del 键也可快速删除宏。

2.3.3　编辑宏

本章前面演示了在 Excel 2003 和 Excel 2007 中录制宏的方法，录制的宏可对当前工作表的选定单元格区域设置格式。如果要对当前工作簿中的每个工作表都执行这些操作，则需要对宏代码进行编辑，使录制的代码循环执行。下面介绍编辑宏代码的操作：

（1）在 Excel 2007 中打开“销售管理（2007）.xls”工作簿。

（2）在功能区【开发工具】选项卡的【代码】组中，单击【宏】按钮，打开如图 2-19 所示的对话框。

（3）在【宏名】列表框中选择“设置表头格式”，单击对话框右侧的【编辑】按钮，将打开如图 2-22 所示的 VBE 窗口。

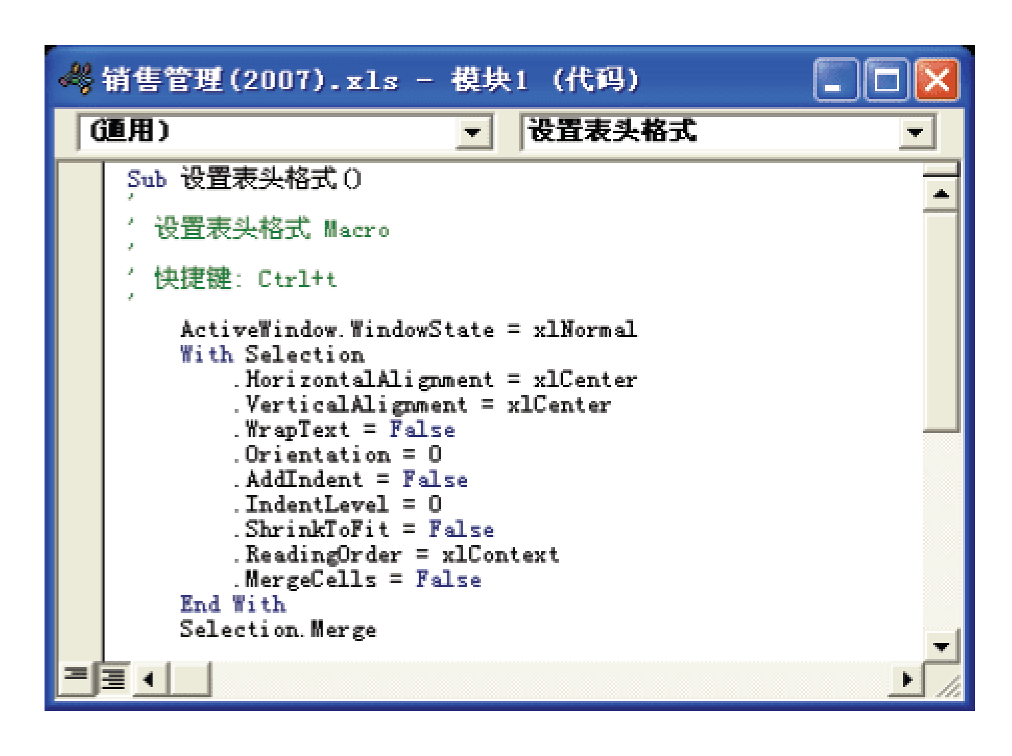


图 2-22　宏代码

（4）在 VBE 窗口中可看到“设置表头格式”的 VBA 代码如下：

```
Sub 设置表头格式()
'
' 设置表头格式 Macro
'
' 快捷键: Ctrl+t
'
    ActiveWindow.WindowState = xlNormal
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlCenter
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    Selection.Merge
    With Selection.Font
        .Name = "宋体"
        .Size = 20
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ColorIndex = xlAutomatic
        .TintAndShade = 0
        .ThemeFont = xlThemeFontNone
    End With
    Selection.Font.Bold = True
    ActiveWindow.WindowState = xlNormal
End Sub
```

（5）从上面的宏代码中可以看出，按要求只需要设置所选单元格区域的字体、字号和粗体即可，但 Excel 录制宏时记录了其他有关字体设置的状态，例如，删除线、阴影和下划线等。可以对上面的代码进行优化，删除多余的代码，最后得到如下所示的代码。

```
Sub 设置表头格式_修改()
    ActiveWindow.WindowState = xlNormal
    With Selection
        .HorizontalAlignment = xlCenter '设置水平居中
        .VerticalAlignment = xlCenter   '设置垂直居中
    End With
    Selection.Merge                     '合并单元格
    With Selection.Font
        .Name = "宋体"                  '设置字体
```

```
        .Size = 20                              '设置字号
    End With
    Selection.Font.Bold = True                  '设置粗体
    ActiveWindow.WindowState = xlNormal
End Sub
```

提示：为了和未修改的代码进行对比，将宏“设置表头格式”的代码复制一份，并改名为“设置表头格式_修改”。

（6）以上代码中删除了不需要的 VBA 代码，执行该宏只能对选中的单元格区域设置格式。如果需要该宏对当前工作簿中所有工作表的第 1 行进行设置，则还需添加代码。修改后的代码如下：

```
Sub 设置表头格式_修改1()
    Dim sh As Worksheet, c As Long                              '声明变量
    ActiveWindow.WindowState = xlNormal
    For Each sh In Worksheets                                   '循环处理所有工
                                                                 作表
        sh.Activate
        c = sh.Range("A2").End(xlToRight).Column                '查找第 2 行最右
                                                                 侧单元格的列号
        If c < 255 Then
            sh.Range(sh.Cells(1, 1), sh.Cells(1, c)).Select     '选中与第 2 行数
                                                                 据等宽的区域
            With Selection
                .HorizontalAlignment = xlCenter                 '设置水平居中
                .VerticalAlignment = xlCenter                   '设置垂直居中
            End With
            Selection.Merge                                     '合并单元格
            With Selection.Font
                .Name = "宋体"                                  '设置字体
                .Size = 20                                      '设置字号
            End With
            Selection.Font.Bold = True                          '设置粗体
        End If
    Next
    ActiveWindow.WindowState = xlNormal
End Sub
```

以上代码对工作簿中的每张工作表进行相同的操作，即查找第 2 列最右侧的数据列，得到第 1 行需要合并的单元格区域，接着选中第 1 行需要合并的区域，最后执行合并、设置字体等操作。

提示：有关 VBA 各命令的意义，这里不做详细介绍，读者学习完本书后面的内容后，就可理解以上代码的含义。

（7）执行以上宏，即可将当前工作簿中所有工作表的第 1 行都进行表头格式设置。

2.4 运 行 宏

创建一个宏之后，就可以在工作簿中反复调用宏进行重复的工作。在 Excel 2007 中，运行宏的方法有很多。本节介绍几种常用的运行宏的方法。

2.4.1 使用快捷键运行宏

在录制宏时，用户可以为每个宏指定一个快捷键。如果是在 VBE 环境中用 VBA 代码编写的宏，也可通过【宏选项】对话框为其指定一个快捷键。

快捷键一般由 Ctrl 键加上一个字母组成。通过快捷键运行宏是最方便的方法，具体操作步骤如下：

（1）打开包含宏的工作簿。

（2）做好执行宏之前的准备工作（如选择单元格区域等），按快捷键 Ctrl+字母便可运行对应的宏。

2.4.2 使用【宏】对话框运行宏

在【宏】对话框有一个【执行】按钮，通过该按钮可运行宏，具体的操作步骤如下：

（1）在 Excel 2007 中打开包含宏的工作簿。

（2）在工作表中拖动选中第 1 行的单元格区域。

（3）在【开发工具】选项卡上的【代码】组中，单击【宏】按钮，打开后如图 2-23 所示。

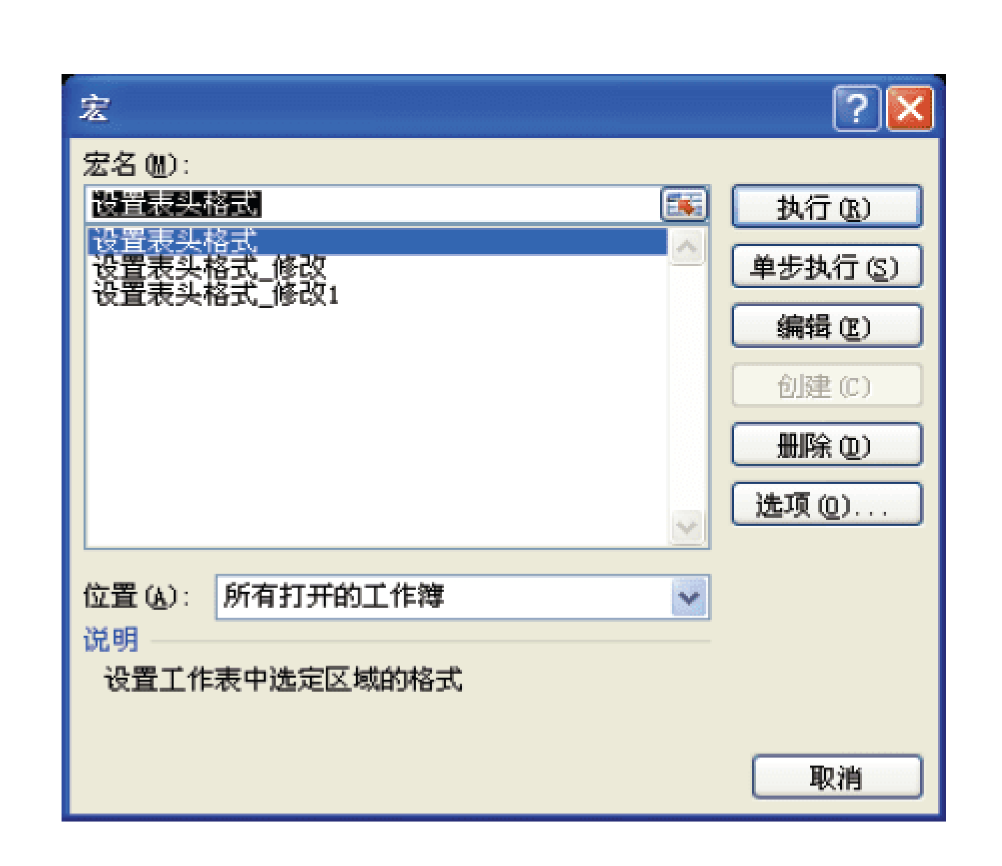


图 2-23　执行宏

（4）在【宏】对话框的【宏名】列表框中，单击选择要运行的宏“设置表头格式”，则在对话框的下方将显示该宏的说明信息。

（5）单击对话框右侧的【执行】按钮即可执行选中的宏。

🔔提示：在执行宏时，按 Esc 键可以中断宏的执行。

2.4.3　使用工具栏运行宏

这种方式只适合于 Excel 2003 及其以前版本。

在 Excel 2003 及其以前版本中，工具栏中汇集了常用命令按钮。用户还可对工具栏进行自定义，甚至将录制的宏添加到工具栏中，以方便用户快速执行宏。使用工具栏执行宏的操作方法如下：

（1）在 Excel 2003 中打开工作簿。

（2）单击主菜单【工具】|【自定义】命令，打开【自定义】对话框，如图 2-24 所示。

（3）在对话框中单击【工具栏】选项卡，如图 2-25 所示。

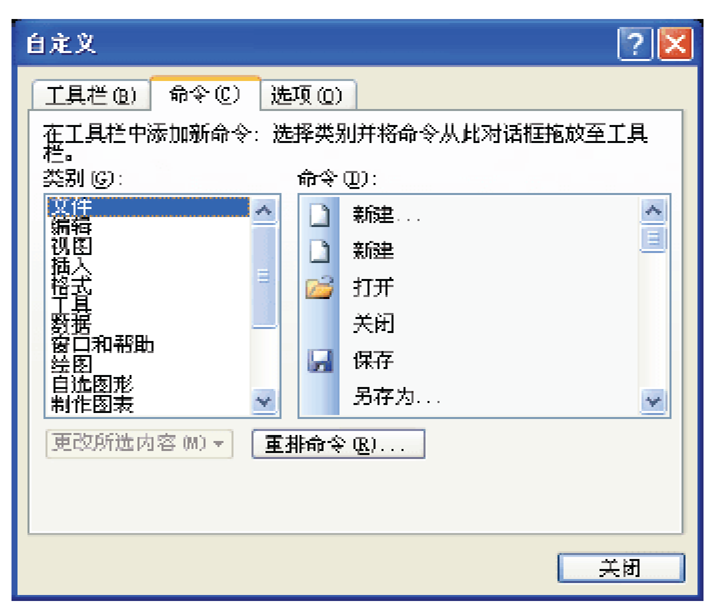



图 2-24 【自定义】对话框

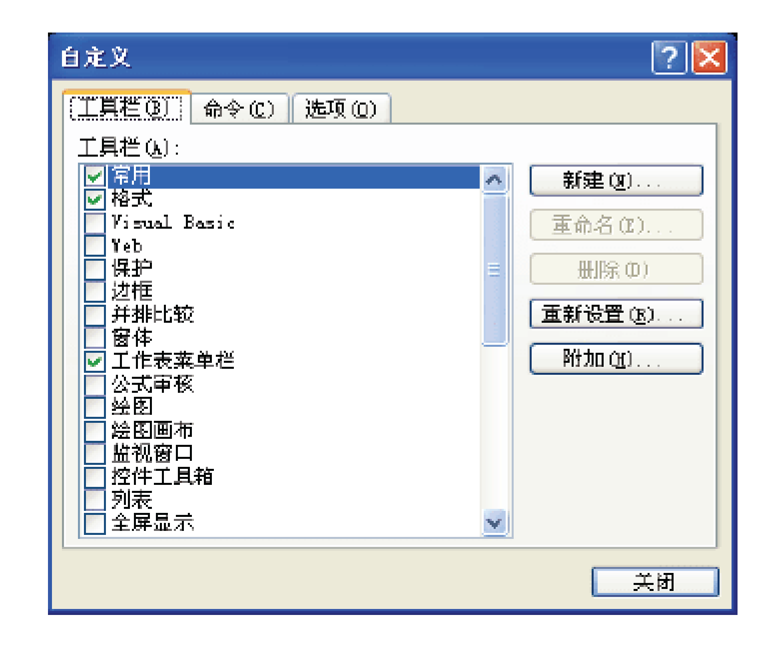


图 2-25 【工具栏】选项卡

（4）在图 2-25 所示对话框中单击【新建】按钮，将打开【新建工具栏】对话框，如图 2-26 所示。

（5）在【工具栏名称】对话框中输入“自定义工具栏”，单击【确定】按钮将创建一个工具栏，如图 2-27 所示。同时在【自定义】对话框中将显示新增加工具栏的名称，如图 2-28 所示。

🔔提示：新创建的工具栏没有任何按钮，是一个空白工具栏。

（6）在【自定义】对话框中单击【命令】选项卡，在【类别】列表框中查找并单击【宏】，如图 2-29 所示。

（7）拖动【自定义】对话框中的【自定义按钮】到新创建的空白工具栏，如图 2-30 所示。



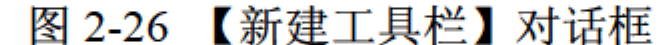

图 2-26 【新建工具栏】对话框



图 2-27　新创建的空白工具栏




图 2-28　工具栏列表

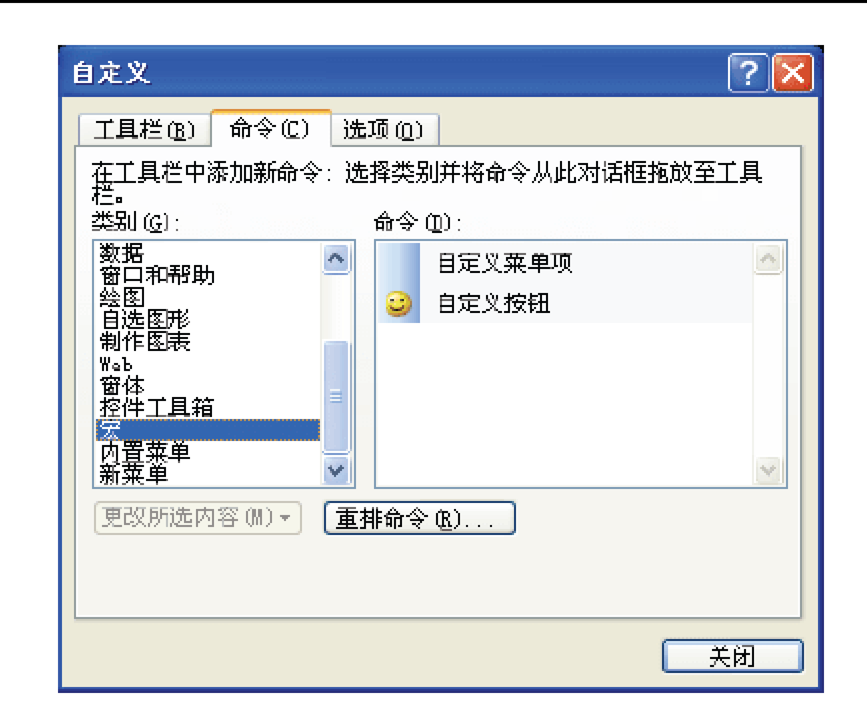


图 2-29　【命令】选项卡

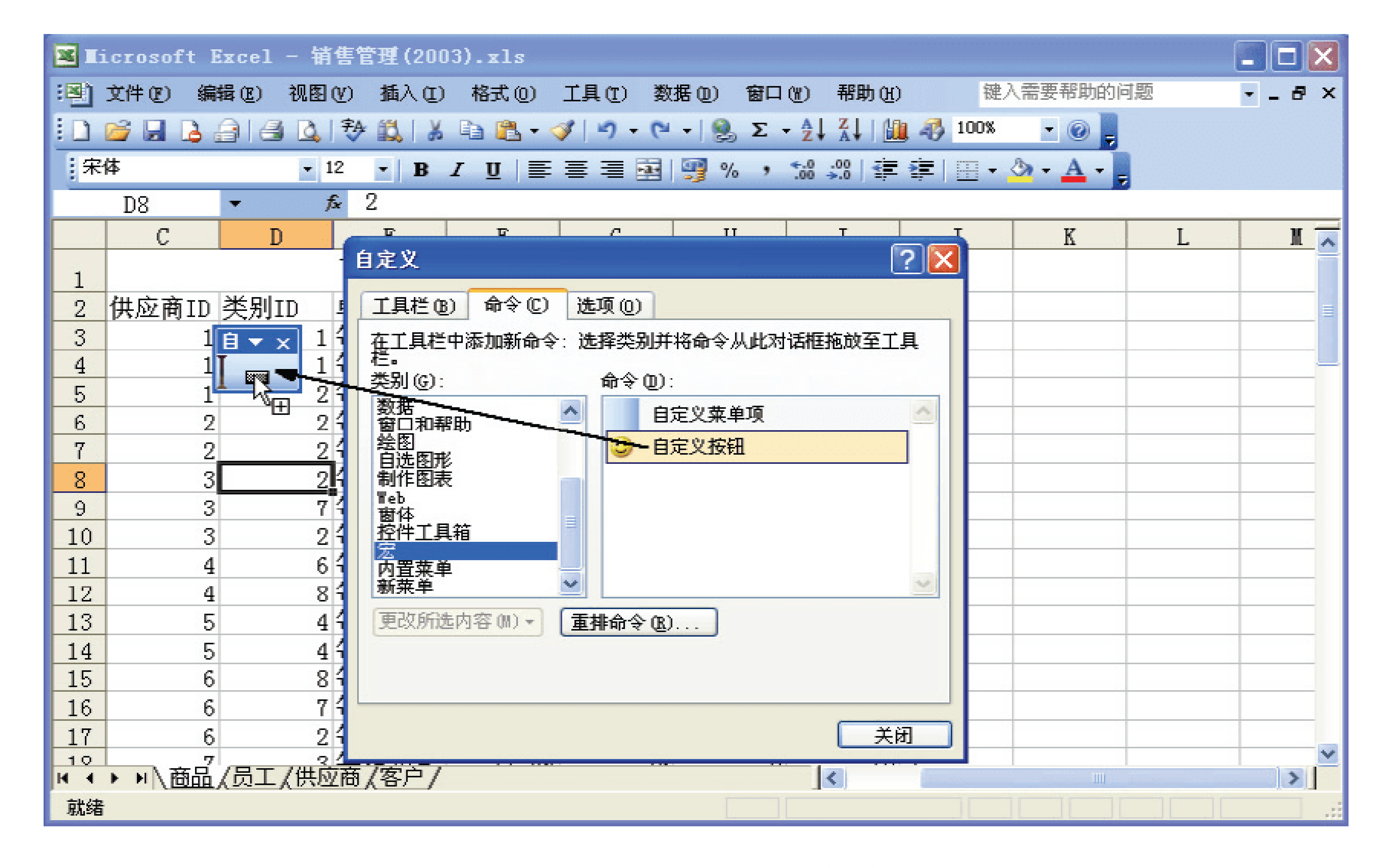


图 2-30　拖动到工具栏中

（8）在新建的工具栏中可以看到新添加的按钮，右击该按钮，弹出快捷菜单，如图 2-31 所示。在快捷菜单的【命令】文本框中输入新建按钮的名称“设置表头格式”。

注意：*必须确保【自定义】对话框处于打开的状态，才能对工具栏中的按钮进行修改。*

（9）再次右击工具栏中的该按钮，在弹出的快捷菜单中单击【更改按钮图像】命令，在弹出的图像列表中选择一个图标，如图 2-32 所示。

（10）再次右击工具栏中的该按钮，在弹出的快捷菜单中单击【指定宏】命令，将打开【指令宏】对话框，如图 2-33 所示。

（11）在【宏名】列表框中选择“设置表头格式”，单击【确定】按钮，即可为工具栏中的按钮指定调用的宏。

（12）单击【自定义】对话框中的【关闭】按钮，完成工具栏的自定义操作。最后得到的新建工具栏如图 2-34 所示。单击工具栏中的按钮即可运行指定的宏。

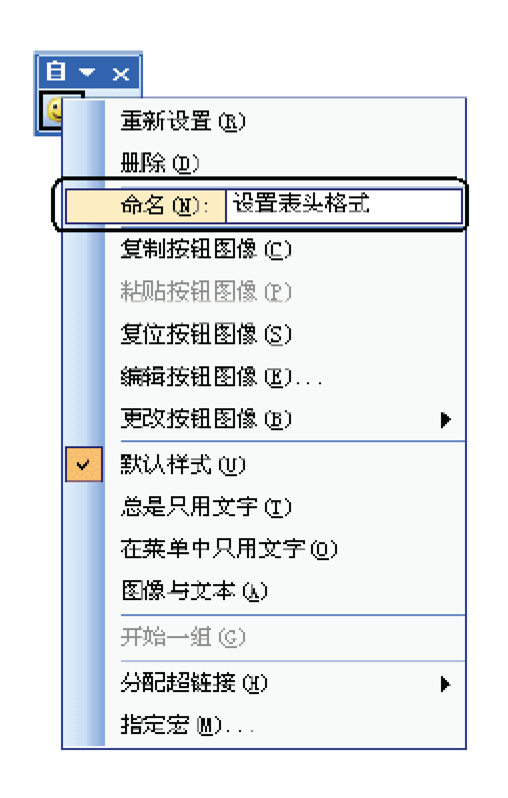


图 2-31　设置按钮提示文本

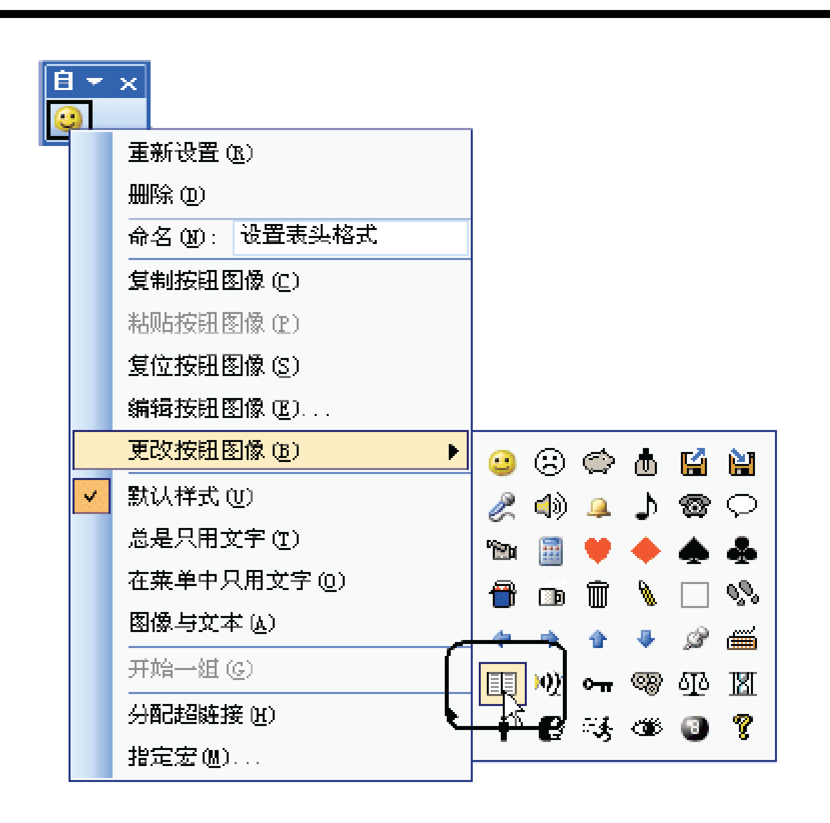


图 2-32　设置按钮图标

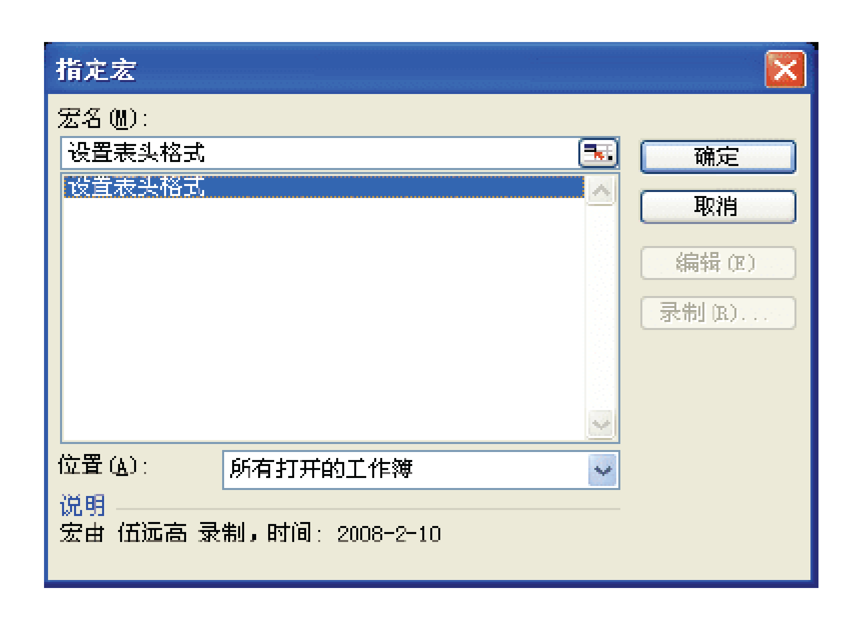


图 2-33　【指定宏】对话框

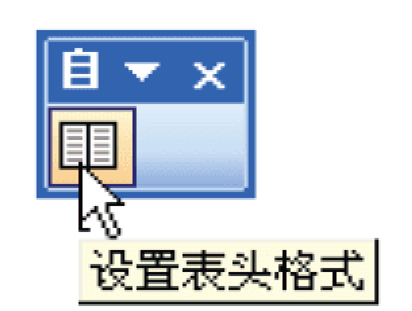


图 2-34　自定义工具栏

技巧：以上操作是在 Excel 环境中进行的，在学完本书相关知识后，读者还可以通过编写 VBA 代码创建自定义工具栏，并为工具栏添加按钮、指定宏。

要删除自定义工具按钮，先打开【自定义】对话框，将工具栏中要删除的按钮拖到【自定义】对话框中，松开鼠标即可。

2.4.4　使用菜单栏运行宏

这种方式只适合于 Excel 2003 及其以前版本。

与使用工具栏运行宏的方法类似，使用菜单栏运行宏的操作也需要通过【自定义】对话框来设置，具体操作步骤以如下：

（1）在 Excel 2003 中打开工作簿。

（2）单击主菜单【工具】|【自定义】命令，打开【自定义】对话框。

（3）在该对话框的【类别】列表框中选择【新菜单】，如图 2-35 所示。

（4）拖动【自定义】对话框中的“新菜单”到 Excel 的菜单栏，如图 2-36 所示。

（5）再次拖动【自定义】对话框中的“新菜单”到 Excel 的菜单栏，在上步创建的“新菜单”处暂停，将打开空的下拉菜单，然后将鼠标拖动到空白下拉菜单中，即可创建一个下拉菜单项，如图 2-37 所示。

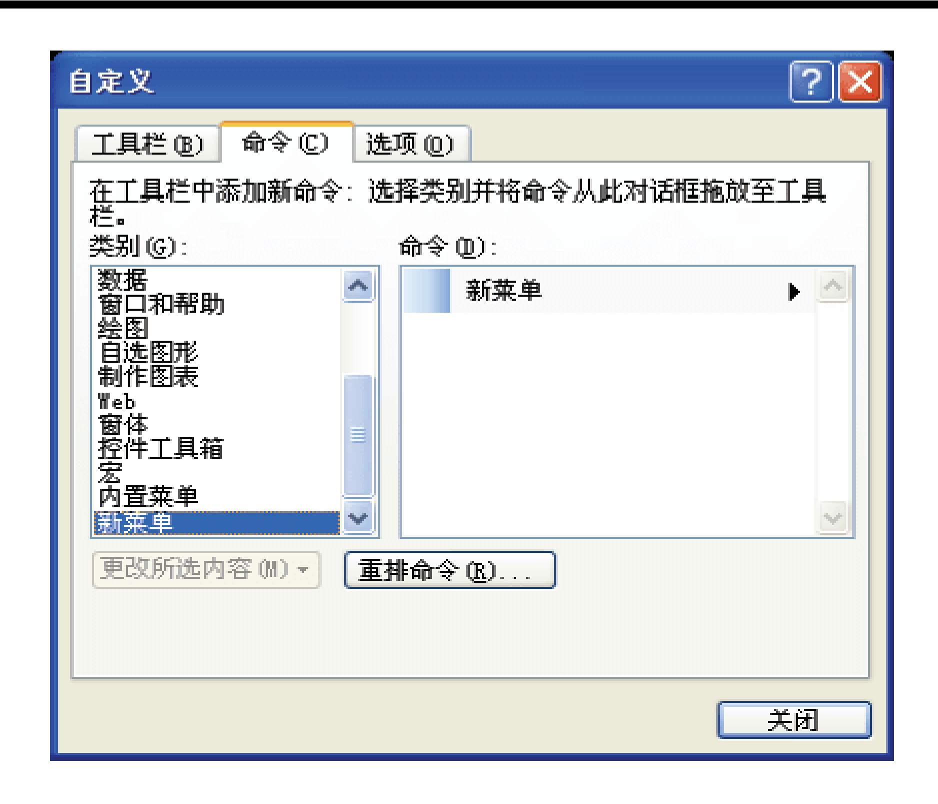


图 2-35 【自定义】对话框中的【新菜单】

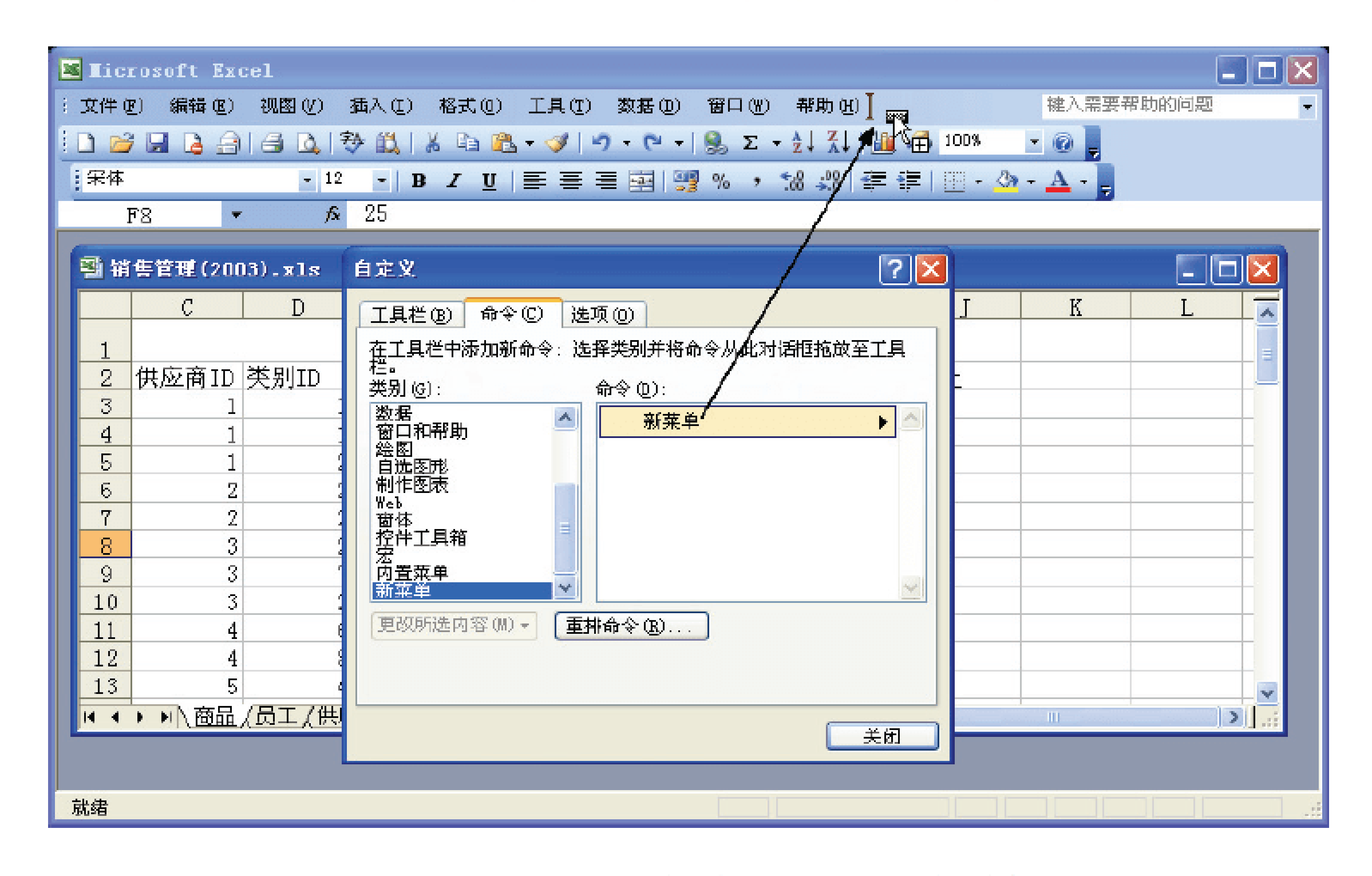


图 2-36 拖动“新菜单”到 Excel 菜单栏

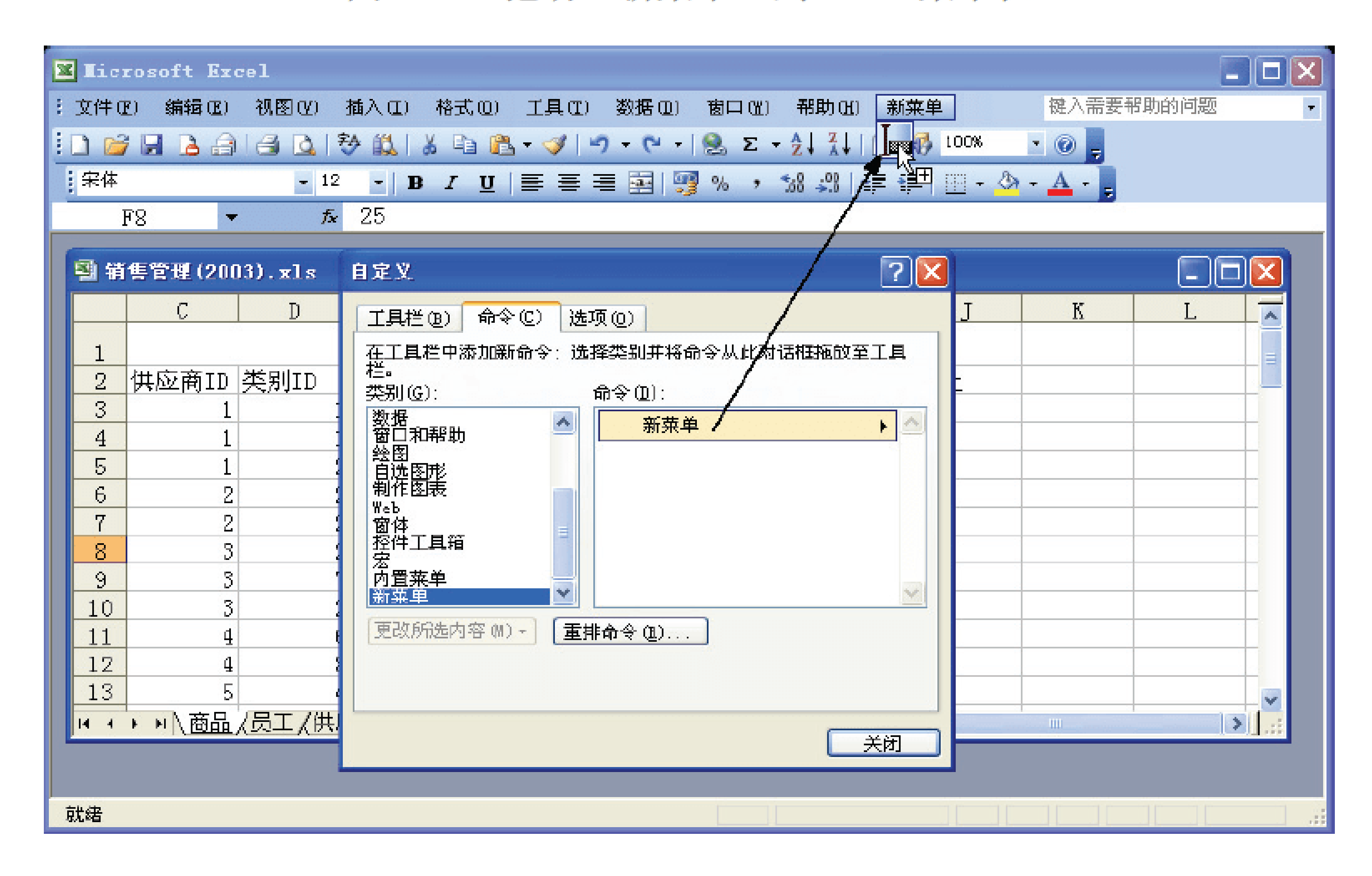


图 2-37 拖动“新菜单”到下拉菜单中

（6）确保【自定义】窗体处于打开状态，单击主菜单中的【新菜单】命令打开下拉菜单，在下拉菜单中右击【新菜单】选项，将弹出快捷菜单，如图 2-38 所示。

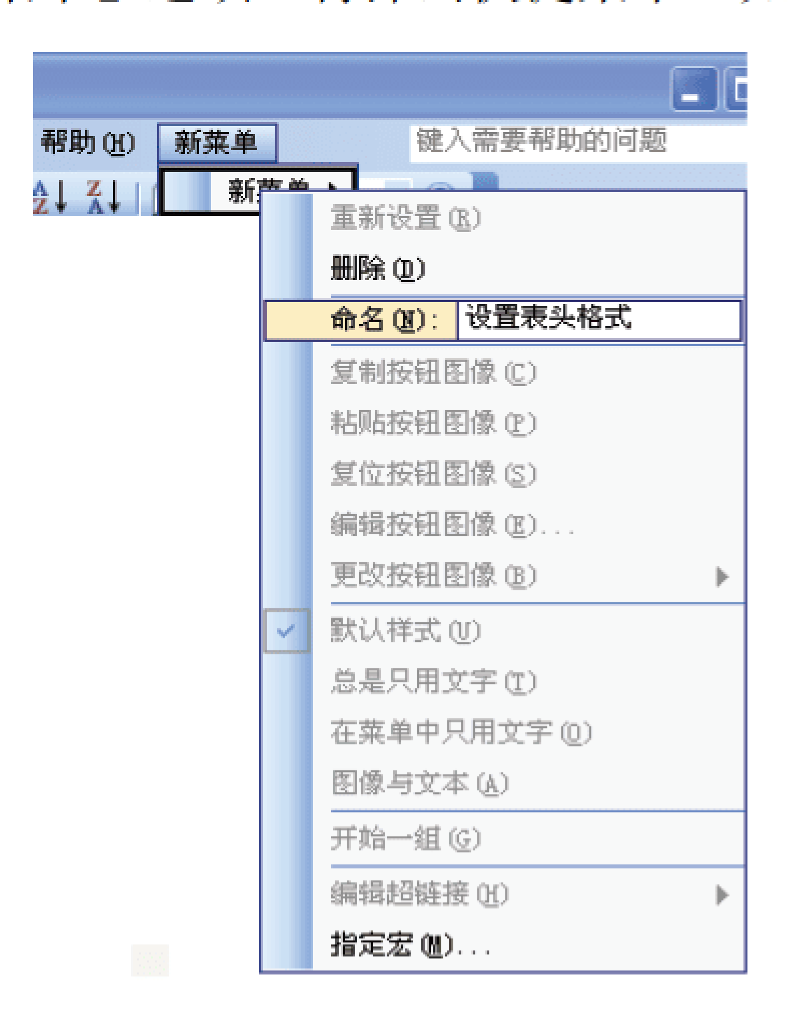


图 2-38　设置菜单文字

（7）在快捷菜单的【命名】文本框中输入菜单的名称“设置表头格式”，如图 2-38 所示。

（8）与自定义工具栏按钮类似，再次右击下拉菜单，在弹出的快捷菜单中选择【指定宏】命令，为下拉菜单项指定执行的宏。

技巧： 要删除自定义菜单，先打开【自定义】对话框，将菜单栏中要删除的菜单项拖到【自定义】对话框中，再松开鼠标即可。

2.4.5　使用快速工具栏运行宏

这种方式适合 Excel 2007 使用。

在 Excel 2007 中，没有了菜单和工具栏，只有快速工具栏与以往版本的工具栏相似。可以在快速工具栏中添加命令按钮，用来快速执行宏，具体操作如下：

（1）在 Excel 2007 中打开工作簿。

（2）单击【Office 按钮】，打开下拉菜单，单击下方的【Excel 选项】按钮，弹出【Excel 选项】对话框。

（3）在该对话框中单击左侧的【自定义】选项卡，显示如图 2-39 所示的界面。在该对话框中，可对自定义快速访问工具栏进行自定义。

（4）在【从下列位置选择命令】下拉列表中，选择【宏】选项，下方的列表框中将显示当前的宏名。

（5）选中一个宏名（如“设置表头格式”），单击【添加】按钮将其添加到自定义快速访问工具栏中，如图 2-40 所示。

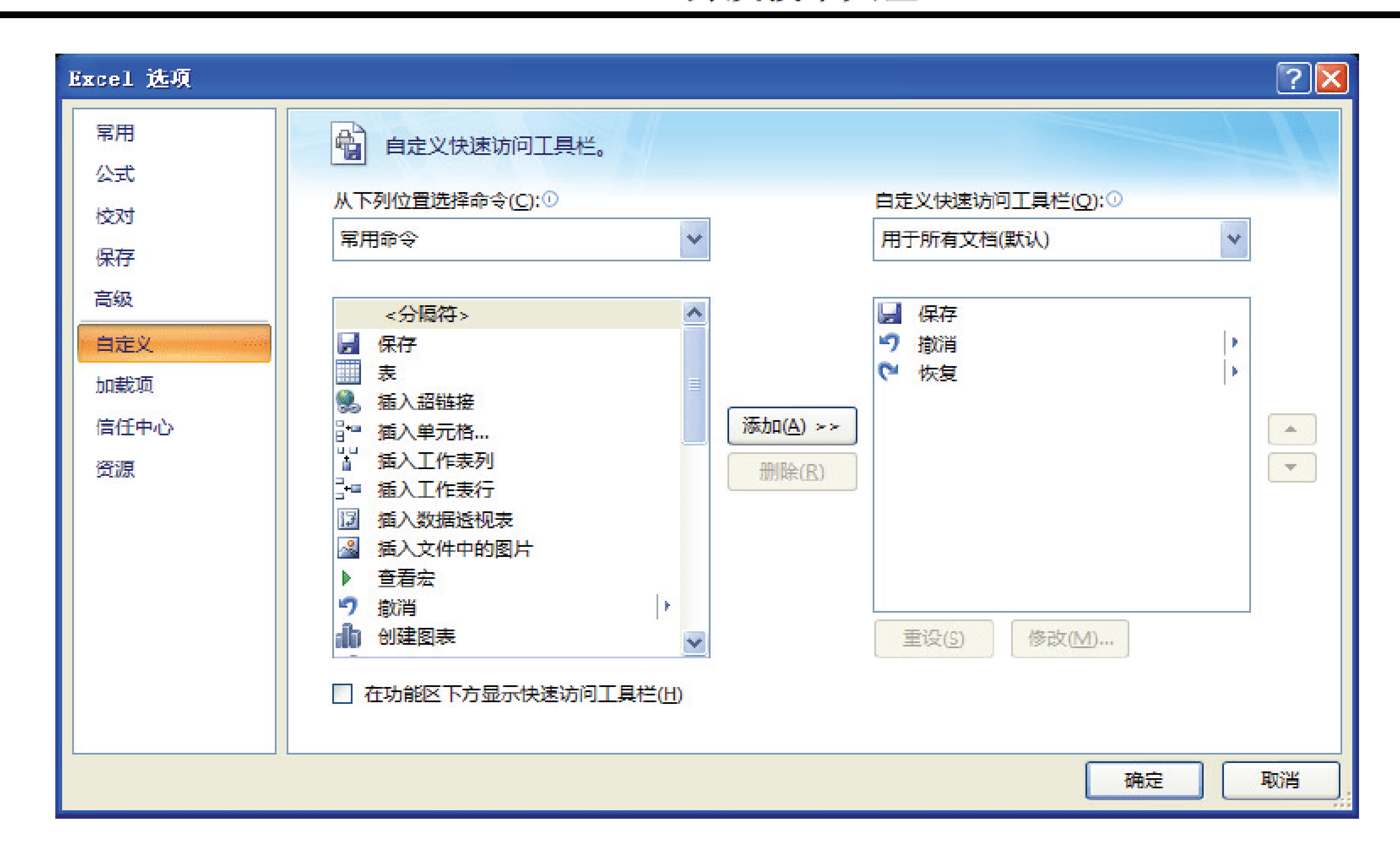


图 2-39 【自定义】选项卡

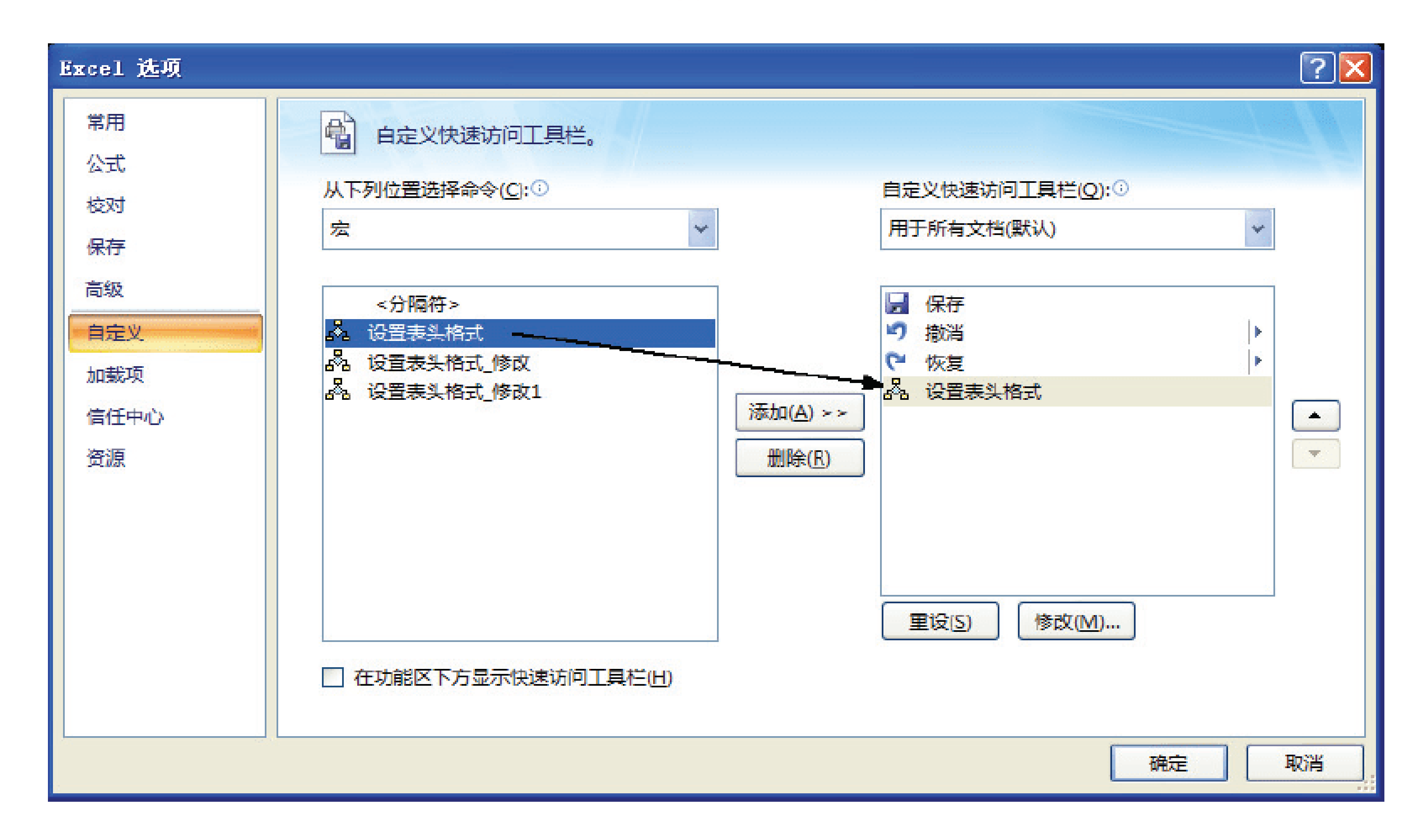


图 2-40 添加按钮到自定义快速访问工具栏

（6）在右侧的列表框中选择刚添加的宏，单击下方的【修改】按钮，在弹出的对话框中，可修改按钮的图标和显示名称，如图 2-41 所示。



图 2-41 【修改按钮】对话框

（7）单击【确定】按钮返回【Excel 选项】对话框，再单击【确定】按钮完成快速工具栏命令按钮的添加操作。此时快速工具栏上将添加一个【设置表头格式】命令按钮，如图 2-42 所示。

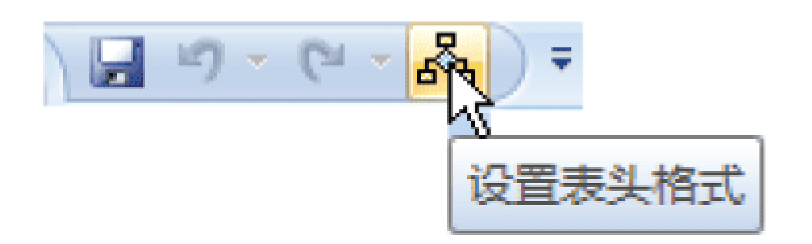


图 2-42　【设置表头格式】按钮

（8）在快速访问工具栏上，单击该按钮即可执行宏。

2.4.6　通过按钮运行宏

在 Excel 的工作表中可插入按钮，用户为插入的按钮指定宏，当单击按钮时就可运行宏。具体操作方法如下：

（1）在 Excel 2007 中打开工作簿。

（2）在功能区【开发工具】选项卡的【控件】组中单击【插入】按钮，弹出如图 2-43 所示的【表单控件】面板。

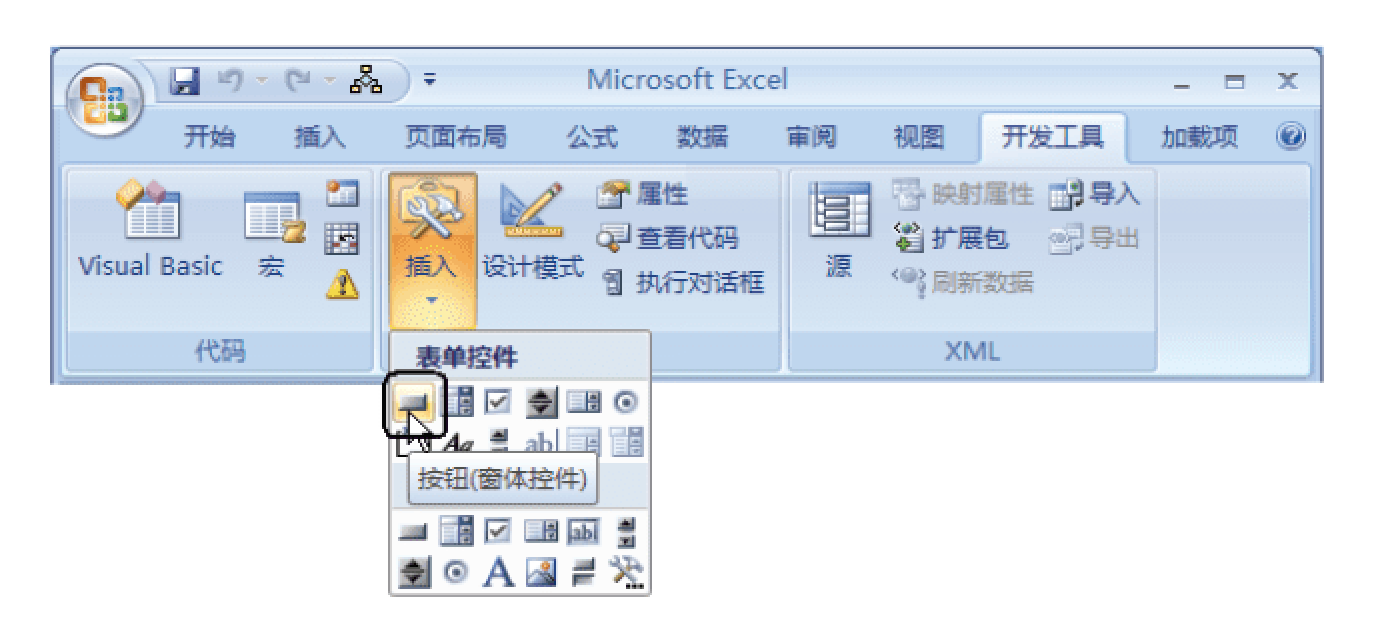


图 2-43　表单控件

（3）在控件面板中单击【按钮】控件，在表格中单击（或拖动鼠标）插入一个按钮，如图 2-44 所示。



	A	B	C	D	E	F	G	H	I	J
1	商品信息表									
2	产品ID	产品名称	供应	类别	单位数量	单价	库存量	订购量	再订购量	中止
3	1	苹果汁	1	1	每箱24瓶	18	39	0	10	是
4	2	牛奶	1	1	每箱24瓶	19	17	40	25	否
5	3	蕃茄酱	1	2	每箱12瓶	10	13	70	25	否
6	4	盐	2	2	每箱12瓶	22	53	0	0	否
7	5	麻油	2	2	每箱12瓶	21.35	0	0	0	是
8	6	酱油	3	2	每箱12瓶	25	120	0	25	否
9	7	海鲜粉	3	7	每箱30盒	30	15	0	10	否
10	8	胡椒粉	3	2	每箱30盒	40	6	0	0	否
11	9	鸡	4	6	每袋500克	97	29	0	0	是
12	10	蟹	4	8	每袋500克	31	31	0	0	否

图 2-44　在表格中插入按钮

（4）当松开鼠标时将会弹出如图 2-45 所示的【指定宏】对话框。在【宏名】列表框中选择宏“设置表头格式”，单击【确定】按钮即可为按钮指定宏。

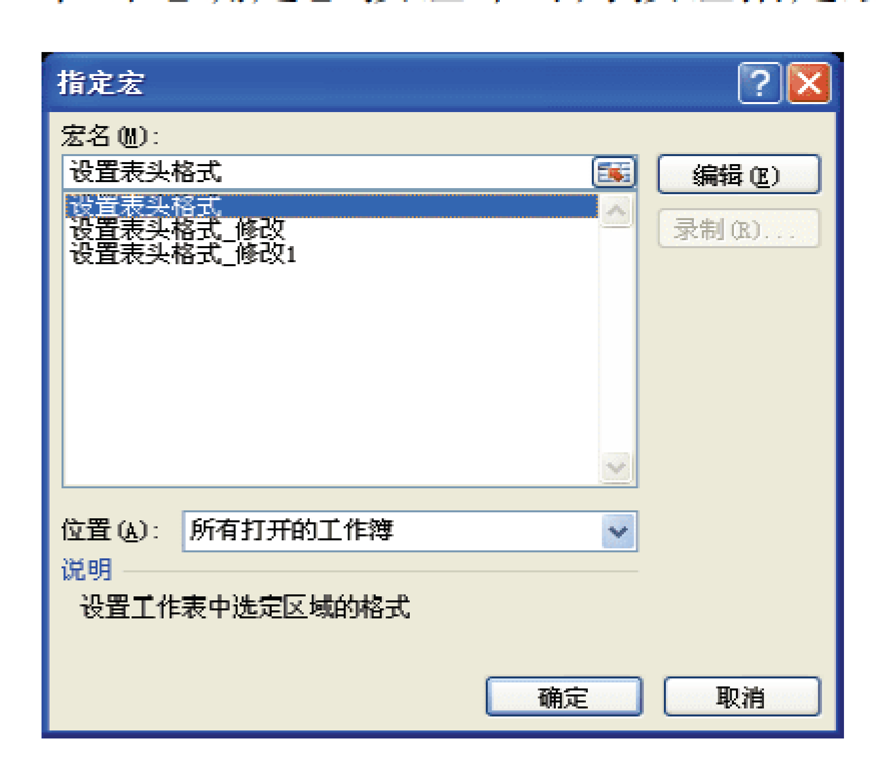
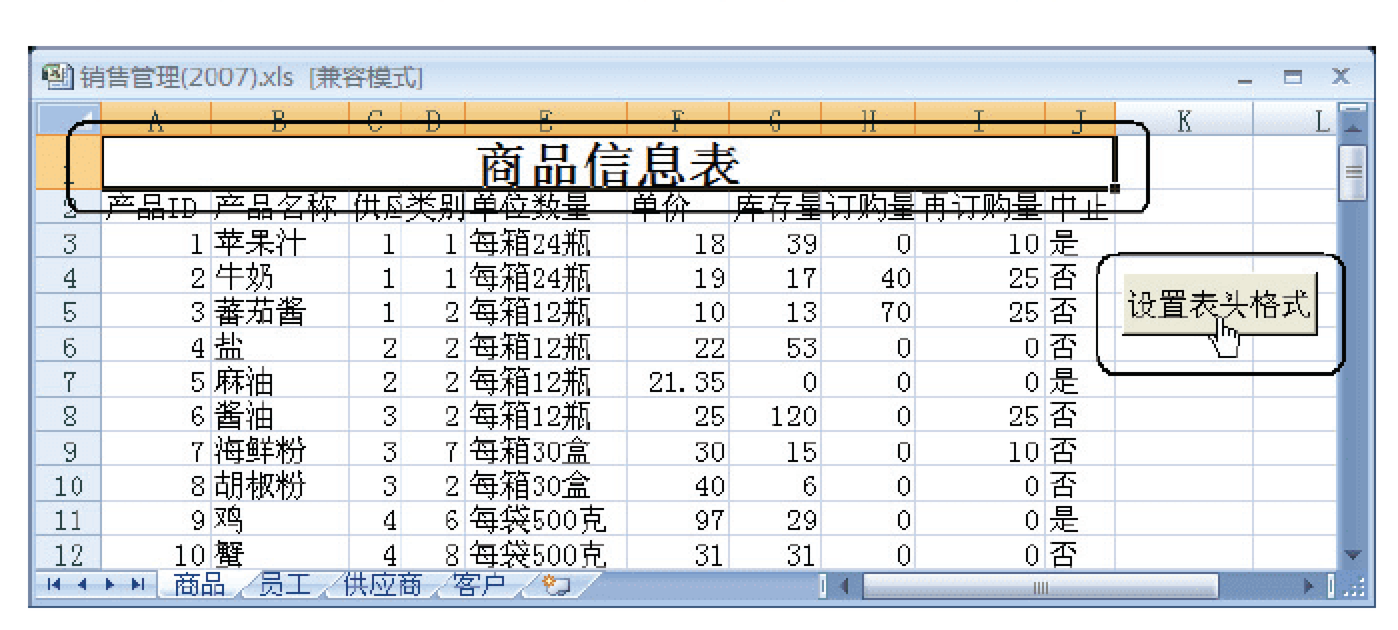


图 2-45 【指定宏】对话框

（5）确保工作表中插入的按钮处于选中状态（按钮周围有 8 个控制点），单击该按钮上的文字，将其修改为“设置表头格式”。然后拖动按钮周围的控制点调整按钮的大小。

（6）在工作表中单击按钮以外的地方，完成宏与按钮的关联设置。

（7）拖动鼠标选择表头部分的单元格区域“A1:J1”，单击工作表中的【设置表头格式】按钮，将运行宏把表头部分设置为相应的格式，如图 2-46 所示。



商品信息表									
产品ID	产品名称	供应	类别	单位数量	单价	库存量	订购量	再订购量	中止
1	苹果汁	1	1	每箱24瓶	18	39	0	10	是
2	牛奶	1	1	每箱24瓶	19	17	40	25	否
3	蕃茄酱	1	2	每箱12瓶	10	13	70	25	否
4	盐	2	2	每箱12瓶	22	53	0	0	否
5	麻油	2	2	每箱12瓶	21.35	0	0	0	是
6	酱油	3	2	每箱12瓶	25	120	0	25	否
7	海鲜粉	3	7	每箱30盒	30	15	0	10	否
8	胡椒粉	3	2	每箱30盒	40	6	0	0	否
9	鸡	4	6	每袋500克	97	29	0	0	是
10	蟹	4	8	每袋500克	31	31	0	0	否

图 2-46 单击【设置表头格式】按钮运行宏

技巧： 在 Excel 工作表中，对于插入的图形对象也可指定宏，其操作方法与给按钮指定宏类似，这里就不再重复了。

2.4.7 打开工作簿自动运行宏

在 Excel 应用程序中，有些宏需要在用户打开工作簿时就自动执行。例如，打开工作簿时显示应用程序的欢迎信息、显示用户登录窗体等。这类宏不需要用户操作就自动运行，在 Excel 中有两种方式来运行：

- 将宏名称设置为 Auto_Open，每次打开包含此宏的工作簿时，该宏都会自动运行。
- 给工作簿对象的 Open 事件编写代码，Open 事件是一个内置的工作簿事件，在每

次打开该工作簿时都运行 Open 事件过程中的代码。

给工作簿对象的事件过程编写代码的内容将在本书第 3 部分中进行详细介绍，本节介绍使用 Auto_Open 宏的方法自动运行宏。假设在用户打开工作簿时要显示一个欢迎信息，可按以下步骤进行操作：

（1）在 Excel 2007 中打开一个工作簿（或新建一个工作簿）。

（2）在功能区【开发工具】选项卡的【代码】组中，单击 Visual Basic 按钮进入 VBE 环境。

（3）在 VBE 中单击主菜单【插入】|【模块】命令，将打开【模块 1】代码窗口，如图 2-47 所示，在代码窗口中输入以下代码：

```
Sub auto_open()
    MsgBox "欢迎使用客户管理系统测试版，" & _
        vbNewLine & "请将使用过程中的问题及时反馈给我们！", _
        vbOKOnly, "欢迎"
End Sub
```

（4）保存并退出当前 Excel 工作簿。

（5）再次打开刚才的工作簿，将会显示如图 2-48 所示的【欢迎】对话框。

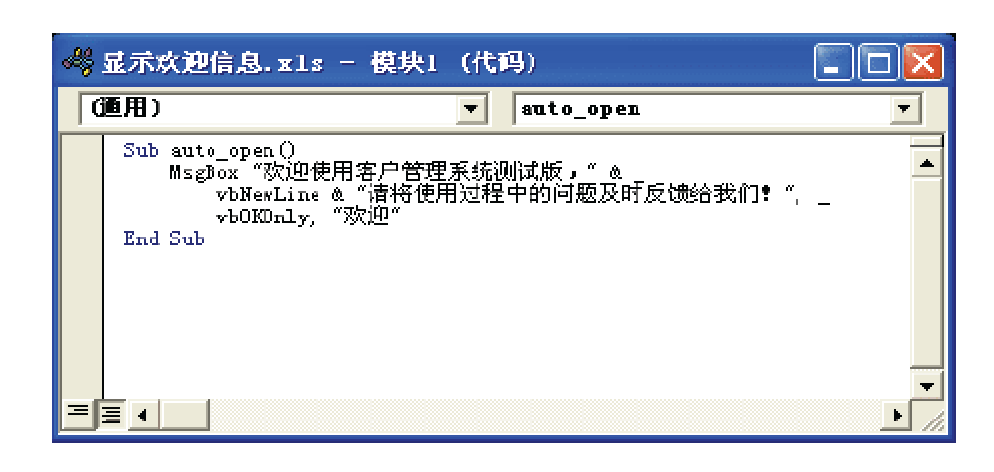


图 2-47 【模块 1】代码窗口

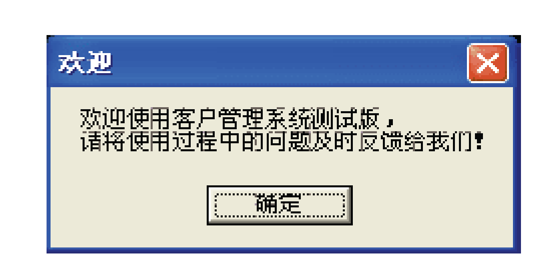


图 2-48 【欢迎】对话框

注意：必须将工作簿的宏安全性设置为【启用所有宏】，打开工作簿时才能自动运行 Auto_Open 宏代码。有关宏安全的内容将在本章后面介绍。

技巧：与 Auto_Open 宏对应的还有一个名为 Auto_Close 的宏，可在关闭工作簿之前完成一些清理操作。

2.5 个人宏工作簿

在录制宏时，将打开如图 2-49 所示的【录制新宏】对话框，在该对话框的【保存在】列表框中可选择宏保存的位置。

录制的宏可保存至以下 3 处：

- 个人宏工作簿；
- 新工作簿；
- 当前工作簿。

如果希望在每次使用 Excel 时都能够使用宏，可选择【个人宏工作簿】列表项。

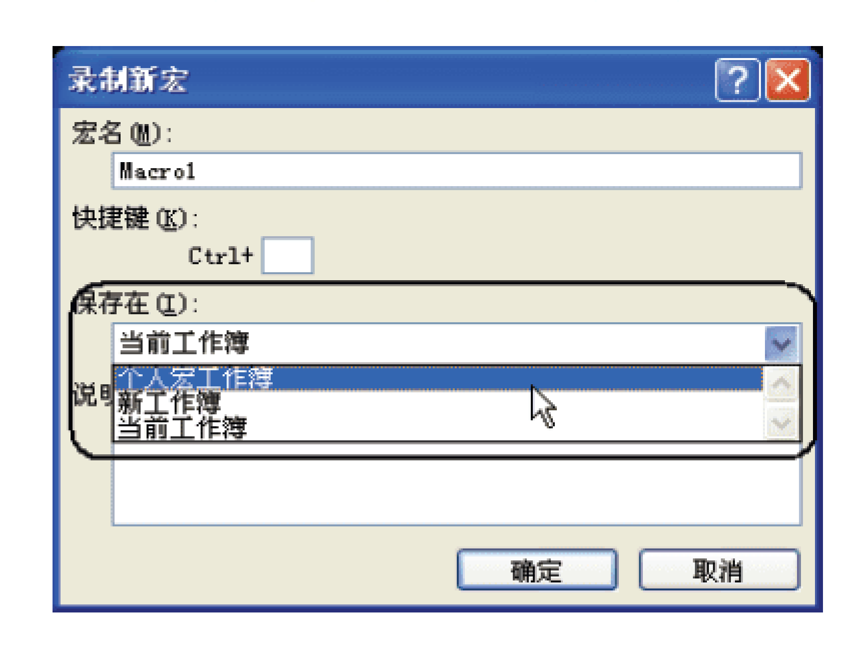


图 2-49 【录制新宏】对话框

2.5.1 了解个人宏工作簿

个人宏工作簿，是为宏而设计的一种特殊的具有自动隐藏特性的工作簿。在图 2-49 所示的对话框中选择“个人宏工作簿”时，如果个人宏工作簿还不存在，Excel 会创建一个，并将宏保存在此工作簿中。如果该文件存在，则每当 Excel 启动时会自动将此文件打开并隐藏在活动工作簿后面。如果要让某个宏在多个工作簿都能使用，那么就应当创建个人宏工作簿，并将宏保存于其中。

Excel 2007 的个人宏工作簿名称为 PERSONAL.XLSB，Excel 2003 及以前版本的个人宏工作簿名称为 PERSONAL.XLS

在 Windows XP 中，个人宏工作簿保存在“C:\Documents and Settings\用户名\Application Data\Microsoft\Excel\XLSTART”文件夹中，以便在每次启动 Excel 时可以自动加载它。在 Windows Vista 中，个人宏工作簿保存在“C:\Users\用户名\Application Data\Microsoft\Excel\XLSTART”文件夹中。

注意：如果当前计算机系统中存在个人宏工作簿，则每次 Excel 启动时会自动将此文件打开并隐藏。

2.5.2 保存宏到个人宏工作簿

将宏保存到个人宏工作簿的操作很简单，常用的有两种方法：

- 在录制宏时，在【录制新宏】对话框中选择【保存在】列表框为“个人宏工作簿”。
- 在 VBE 环境中将宏代码复制到个人宏工作簿中。使用这种方法时，个人宏工作簿必须已经存在。

下面介绍第一种方法的操作步骤：

（1）在 Excel 2007 中打开一个工作簿，选择一个要设置边框线的单元格区域。

（2）在功能区【开发工具】选项卡的【代码】组中，单击【录制宏】按钮，打开【录制新宏】对话框。

（3）在该对话框的【宏名】文本框中输入名称，在【保存在】列表框中选择“个人宏工作簿”，如图 2-50 所示。

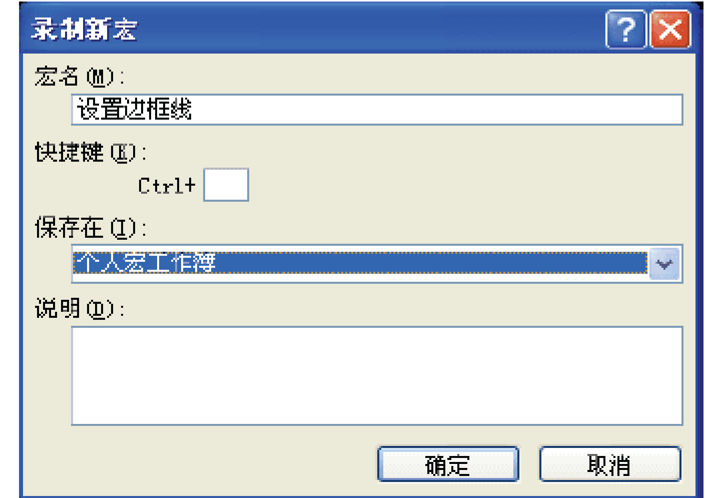


图 2-50　【录制新宏】对话框

（4）单击【确定】按钮，开始录制宏。将功能区切换到【开始】选项卡，单击【字体】组中的【边框线】按钮，为选中区域设置边框线。

（5）将功能区切换到【开发工具】选项卡，单击【代码】组中的【停止录制】按钮 停止录制，完成宏的录制。

（6）在 Excel 环境中，按快捷键 Alt+F11 进入 VBE 环境，可以看到除了打开的工作簿以外，还有一个名为 PERSONAL.XLSB 的工作簿，单击打开其【模块 1】代码窗口，可看到录制的宏代码，如图 2-51 所示。

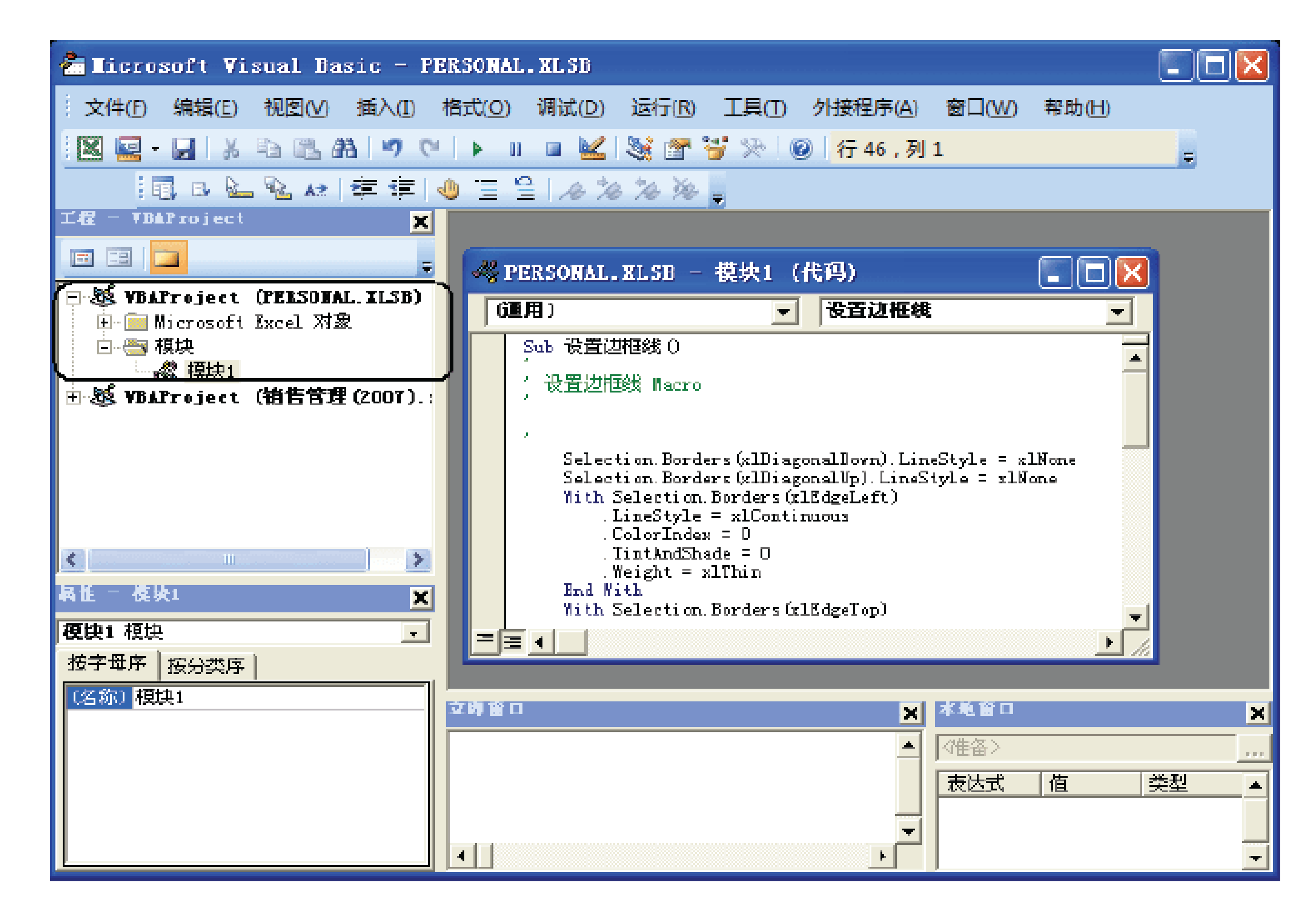


图 2-51　个人宏工作簿代码

2.5.3　管理个人宏工作簿

如果系统中已有个人宏工作簿，在每次打开 Excel 时，该工作簿都将自动打开。

1．显示 / 隐藏个人宏工作簿

在正常情况下，个人宏工作簿处于隐藏状态，在 Excel 环境中看不到该工作簿，而在 Excel VBE 中可看到该工作簿中的代码。

其实，用户也可以在 Excel 环境中显示出个人宏工作簿。具体操作步骤如下：

（1）在 Excel 2007 中，将功能区切换到【视图】选项卡，单击【窗口】组中的【取消隐藏】按钮，如图 2-52 所示。

（2）在弹出的【取消隐藏】对话框中选中 PERSONAL.XLSB，如图 2-53 所示。

（3）单击【确定】按钮，个人宏工作簿即可显示在 Excel 环境中，如图 2-54 所示。

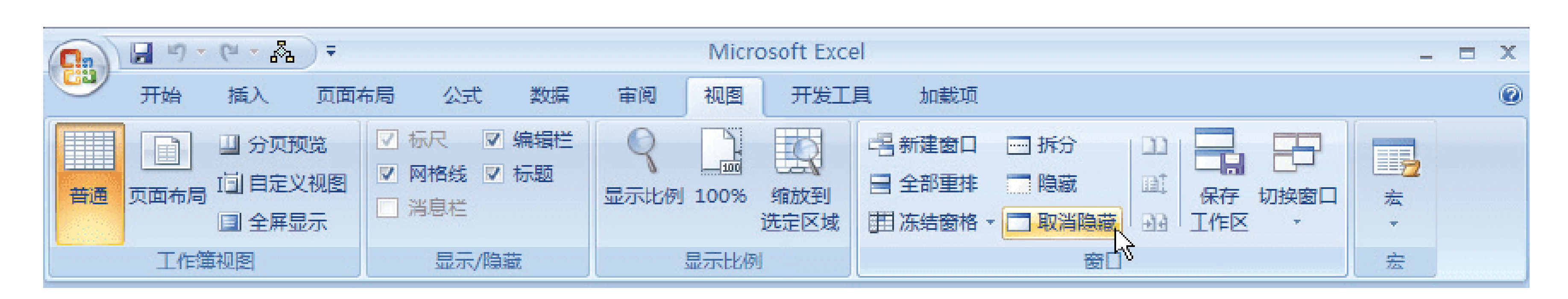


图 2-52　单击【取消隐藏】按钮

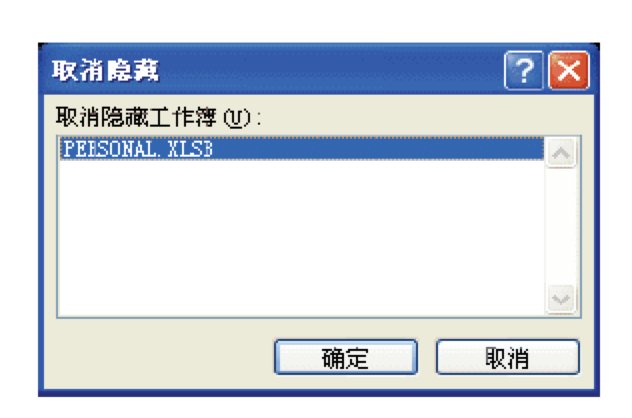




图 2-53　【取消隐藏】对话框

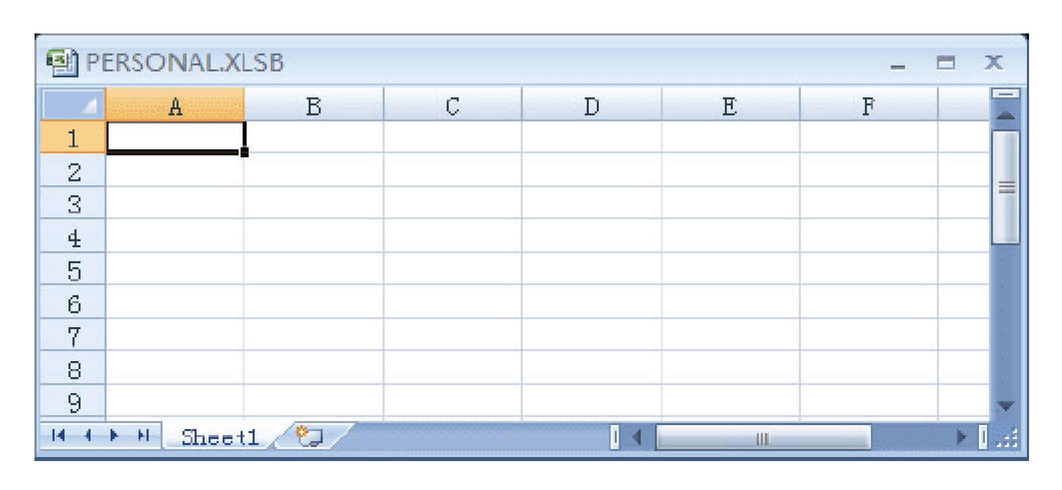


图 2-54　个人宏工作簿

如果要隐藏个人宏工作簿，确认 PERSONAL.XLSB 处于最前面的状态，可以在【视图】选项卡中，单击【窗口】组中的【隐藏】按钮即可。

注意：若显示分辨率过小，【视图】选项卡中的【取消隐藏】按钮和【隐藏】按钮可能将只显示一个图标，而不显示文字。

2. 编辑个人宏工作簿中的宏

开发人员可对个人工作簿中的宏代码进行编辑，其操作方法与 2.3.3 节的方法类似。按快捷键 Alt+F11 进入 VBE 环境后，在【工程】子窗口中双击 PERSONAL.XLSB，展开包含的模块，打开对应的代码窗口，即可对代码进行编辑操作。

2.6 宏的安全性

宏是由 VBA 代码组成的，从 Office 软件支持宏开始，宏病毒也随之出现。许多病毒经过专门设计，可以利用 VBA 代码对系统和数据文件进行恶意操作。因此，宏的安全性越来越受到用户的重视。Excel 2007 提供了更高更细致的宏安全性，能够在大部分情况下杜绝宏病毒对工作簿造成的危害。

2.6.1 打开包含宏的文档

默认情况下，当打开包含有宏的工作簿时，在编辑栏上方将会显示安全警告，如图 2-55 所示。这时宏是被禁用的，如果要启用宏，可按以下步骤操作：

（1）单击安全警告右侧的【选项】按钮，打开【Microsoft office 安全选项】对话框，如图 2-56 所示。

（2）选中【启用此内容】单选按钮后，单击【确定】按钮关闭对话框，工作簿中的宏

将被启用。

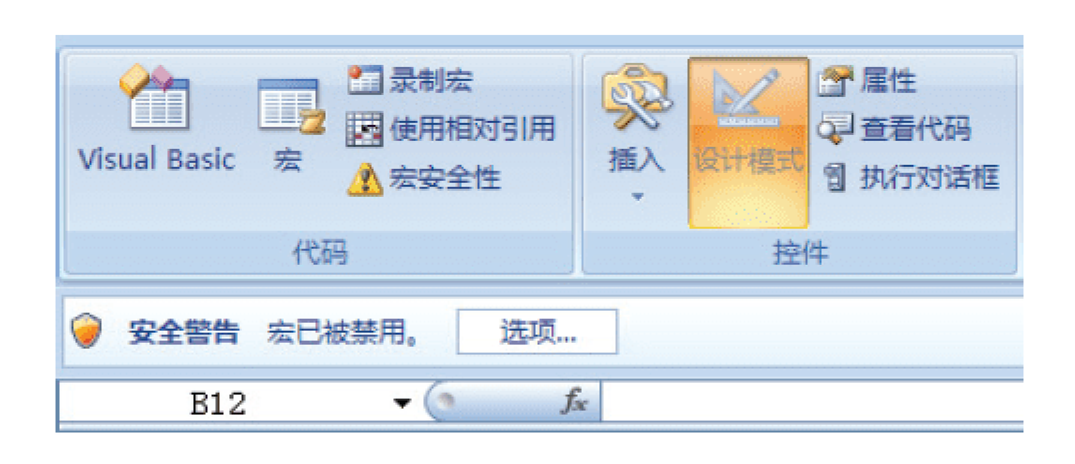


图 2-55　安全警告信息



图 2-56　【Microsoft office 安全选项】对话框

如果不能确定打开工作簿的宏是安全的，可以在出现图 2-55 所示的安全警告信息时，不做任何处理，则工作簿中的宏将不能被执行，从而保证系统和文档的安全。

2.6.2　设置宏的安全性

用户可根据工作环境，对宏的安全性进行设置。设置宏安全性的步骤如下：

（1）单击【Office 按钮】按钮，在下拉菜单中单击下方的【Excel 选项】按钮，弹出【Excel 选项】对话框。

（2）单击左侧的【信任中心】选项卡，对话框显示如图 2-57 所示。

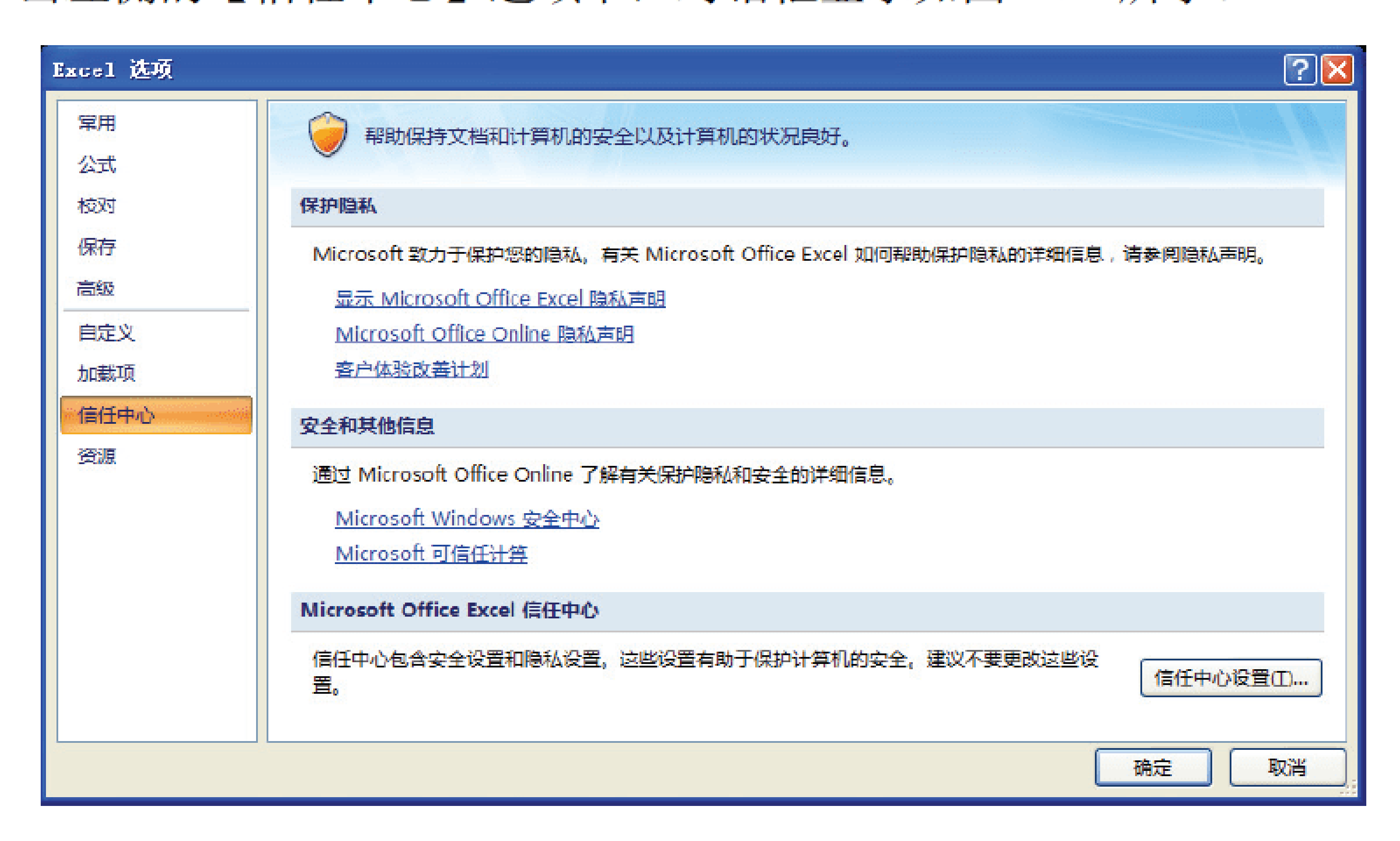


图 2-57　Excel 选项【信任中心】选项卡

（3）单击右下方的【信任中心设置】按钮，弹出【信任中心】对话框。

（4）单击左侧的【宏设置】选项卡，对话框显示如图 2-58 所示。

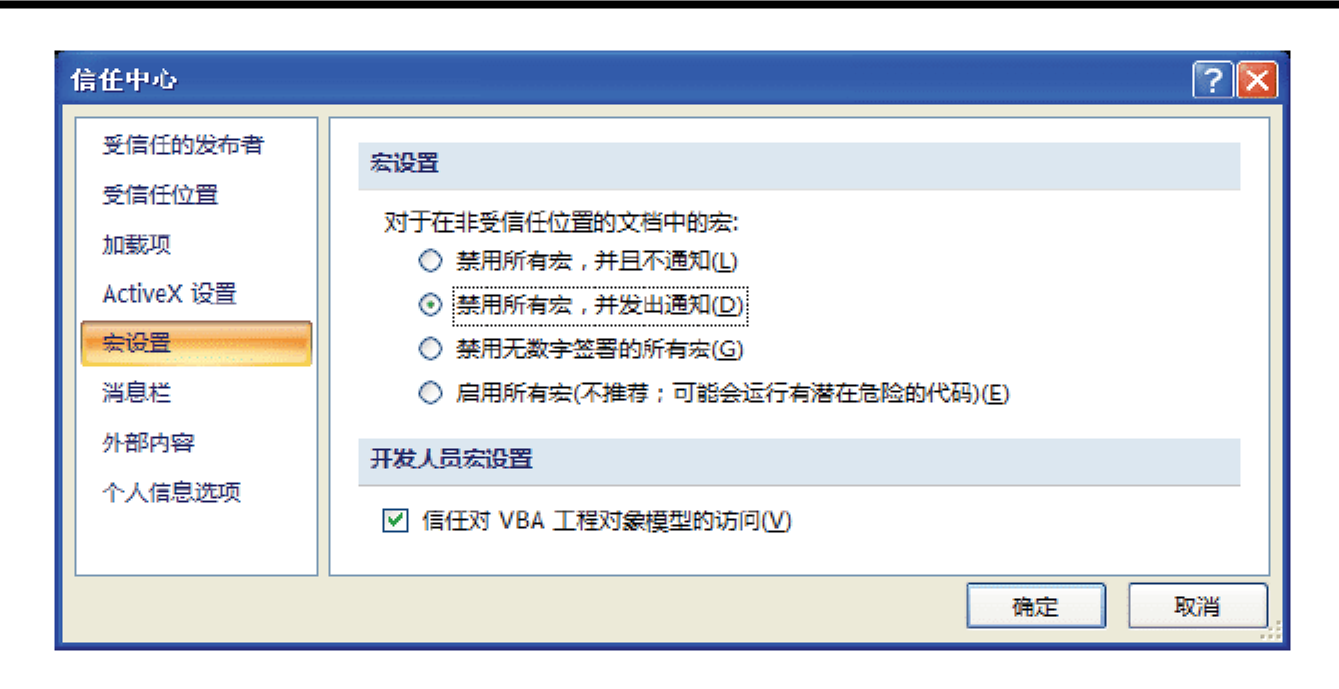


图 2-58 【宏设置】选项卡

注意：在【开发工具】选项卡的【代码】组中，单击【宏安全性】按钮也可打开如图 2-58 所示的【信任中心】对话框。

（5）在【宏设置】选项区域中有以下 4 个选项，其作用分别为：

- 禁用所有宏，并且不通知：如果不信任宏，使用此设置。文档中的所有宏以及有关宏的安全警报都被禁用。如果文档具有信任的未签名的宏，则可以将这些文档放在受信任位置。受信任位置中的文档可直接运行，不会由信任中心安全系统进行检查。
- 禁用所有宏，并发出通知：这是默认设置。如果想禁用宏，但又希望在存在宏的时候收到安全警报，则应使用此选项。这样，可以根据具体情况选择何时启用这些宏。
- 禁用无数字签署的所有宏：此设置与【禁用所有宏，并发出通知】选项相同，但下面这种情况除外：在宏已由受信任的发行者进行了数字签名时，如果信任发行者，则可以运行宏；如果还不信任发行者，将收到通知。这样，可以选择启用那些签名的宏或信任发行者。所有未签名的宏都被禁用，且不发出通知。
- 启用所有宏（不推荐，可能会运行有潜在危险的代码）：可以暂时使用此设置，以便允许运行所有宏。因为此设置容易使计算机受到可能是恶意代码的攻击，所以不建议永久使用。

一般情况下，使用 Excel 2007 的默认设置即可。如果用户使用的工作簿都是自己（或工作组成员）制作的，即可选择最后一项【启用所有宏（不推荐，可能会运行有潜在危险的代码）】，这样在打开工作簿时将不会出现安全警告信息。

第 3 章　Excel VBA 的开发环境

通过在 Excel 中录制的宏，可反复执行一些重复操作，在一定程度上可提高用户的工作效率。

对于录制的宏，有时还需要对录制的代码进行编辑修改。更多的情况是，用户需要直接编写代码，或设计与用户交互的窗体。这些操作都需要在 Excel VBA 的开发环境——VBE 中进行。本章详细介绍 VBE 开发环境的使用。

3.1　VBE 简介

VBE 是编写 VBA 代码的工具，开发人员可在该环境中对录制的宏代码进行修改，或直接编写代码、插入用户窗体等。在第 2 章中编辑宏时曾使用过 VBE 环境。

3.1.1　VBE 概述

VBE 是一个非常友好的开发环境，在 VBE 中编写的代码将成为 Excel 工作簿文件的一部分，例如，在对 Excel 文件进行保存操作时，VBA 的组成部分（如模块、用户窗体等）同时也进行了保存。

VBE 是一个单独的应用程序，拥有独立的操作窗口，可以与 Excel（或其他的 Office 组件，如 Word）无缝地结合。但 VBE 环境不能单独打开，要使用 VBE，必须先运行 Excel（或其他 Office 组件）。

在 Excel 中，使用 VBE 开发环境可以完成以下任务：

- 创建 VBA 过程；
- 创建用户窗体；
- 查看/修改对象属性；
- 调试 VBA 程序。

3.1.2　进入 VBE

在 Excel 2007 中可以有多种方式进入 VBE，常用的方法如下：

- 在功能区【开发工具】选项卡的【代码】组中，单击 Visual Basic 按钮可进入 VBE 环境。
- 在 Excel 中，按快捷键 Alt+F11 可进入 VBE 环境，这是最常用的方法。

- 在功能区【开发工具】选项卡的【代码】组中，单击【宏】按钮，打开【宏】对话框，选中一个宏，单击【编辑】按钮可进入 VBE 环境。
- 右击 Excel 工作表标签，弹出快捷菜单，选择【查看代码】命令可进入 VBE 环境。

3.1.3 VBE 操作界面

VBE 是 Office 与 VB 两种环境的集合体，因此在界面方面继承了 Office 与 VB 的优点，VBE 的默认界面如图 3-1 所示。下面分别介绍 Excel 2007 VBE 界面的各个构成部分。

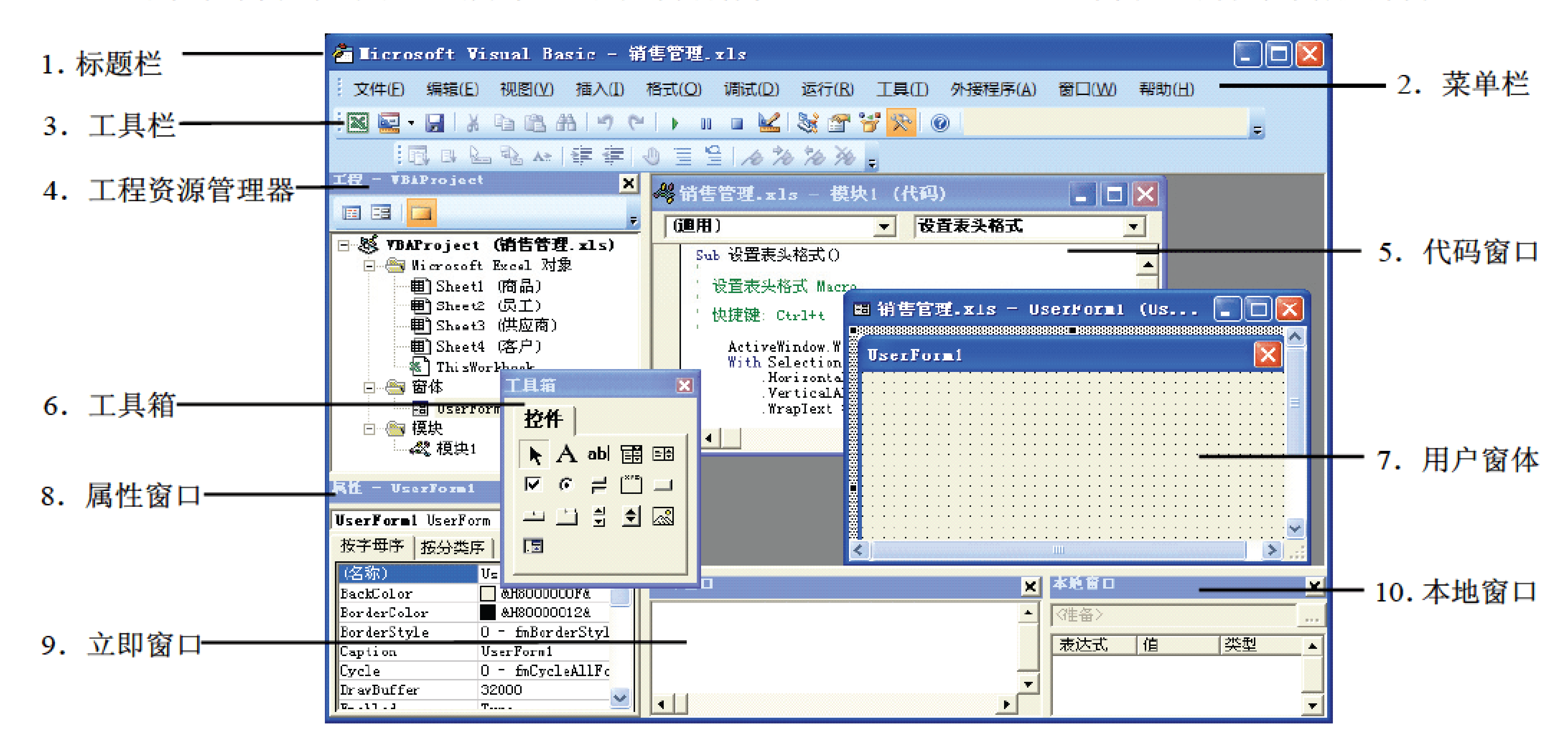


图 3-1 VBE 操作界面

1. 标题栏

标题栏是用来显示打开窗口的标题。标题栏分 3 部分显示对应内容，前面部分显示开发环境名称 Microsoft Visual Basic，中间部分显示窗体名称（如“销售管理.xls”），最后部分为控制 VBE 窗体的按钮（最小化、最大化和关闭按钮），如图 3-2 所示。



图 3-2 标题栏

2. 菜单栏

VBE 的菜单栏与其他 Windows 应用程序的菜单栏相同。VBE 菜单栏包含了绝大多数的命令，用户只需逐项选择菜单即可执行对应的命令。

菜单栏主要包含文件、编辑、视图、插入、格式、调试、运行、工具、外接程序、窗口以及帮助 11 个菜单项，每一个菜单项都含有若干个菜单命令或子菜单，通过单击菜单下的命令或子菜单就可以执行相应的操作，如图 3-3 所示。



图 3-3　菜单栏

菜单栏中各菜单项的功能和作用如下面所述。

- 文件：主要是对文件进行保存、导入、导出和退出操作。
- 编辑：主要是对应用程序代码进行撤销、复制、清除、查找、缩进、凸出等基本代码编辑操作，以及显示属性/方法列表、常数列表、快速列表等。
- 视图：主要是对 VBE 窗口进行隐藏/显示管理，如代码窗口、对象窗口、对象浏览器、立即窗口、本地窗口、监视窗口等。
- 插入：主要是对类模块、过程和文件等进行插入操作。
- 格式：主要是对工作表中添加的控件的位置和大小等进行调整操作。
- 调试：主要是对代码进行编译、调试、监视等操作。
- 运行：主要是对代码进行运行、中断、重新设置和设计模式操作。
- 工具：主要是对 VBE 选项和宏进行管理。
- 外接程序：主要是对外接程序进行管理。
- 窗口：主要是对各窗口的显示方式进行管理。
- 帮助：主要是链接 VB 帮助文件和打开 Web 上的 MSDN 链接。

3．工具栏

VBE 有 6 个工具栏（菜单条和快捷菜单也包含在工具栏中），默认情况下，【标准】工具栏显示在菜单栏的下方，其他工具栏都处于隐藏状态。与 Excel 2003 中的工具栏操作相似，用户可以对工具栏进行移动、停靠、显示、隐藏等操作。

单击主菜单【视图】|【工具栏】|【自定义】命令，将打开如图 3-4 所示的【自定义】对话框，在其中的【工具栏】选项卡中可以选择需要显示的工具栏，也可根据需要新建用户自己的工具栏。

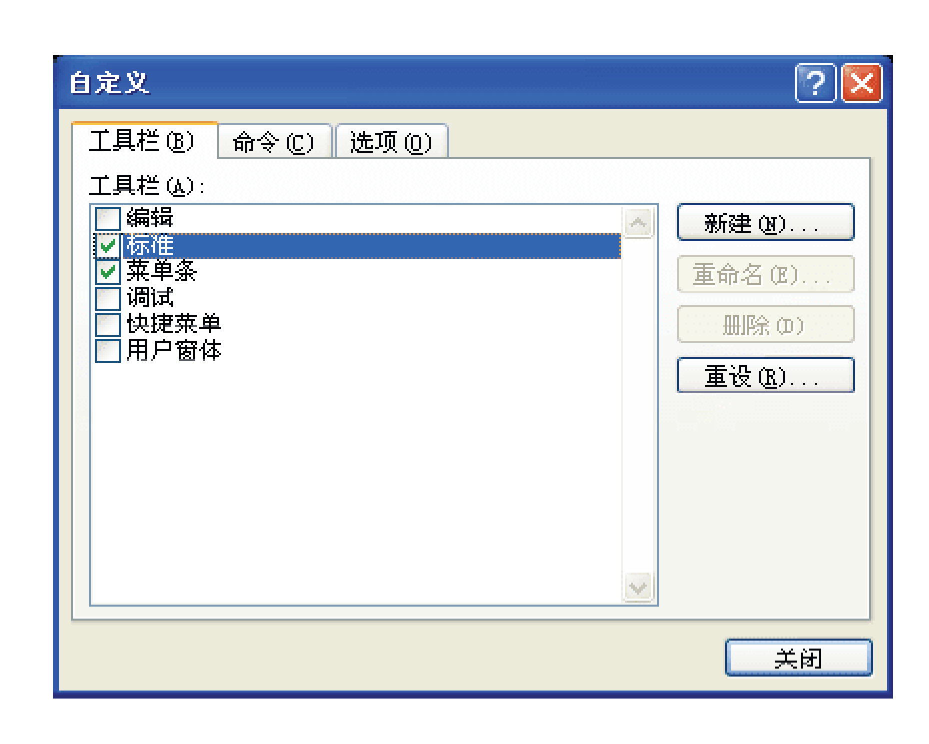


图 3-4 【自定义】对话框

各工具栏的主要作用分别如下所述。

- 【标准】工具栏：主要显示常用的功能按钮，包括视图 Microsoft Excel、插入用户窗体、保存、剪切、复制、查找、撤销、运行子过程/用户窗体、中断、工程资源管理器、设计模式、属性窗口、对象浏览器等。
- 【编辑】工具栏：主要用来对代码进行操作，如对程序代码进行缩进凸出、显示属性/方法列表、显示常数列表、显示快速列表、书签等操作。
- 【调试】工具栏：主要是对代码进行编译、调试、监视、切换断点、逐语句、逐过程等操作。
- 【用户窗体】工具栏：主要是对开发具体窗体控件进行操作，如移至顶层、移至底层、组、取消组、左对齐等。

4. 工程资源管理器

工程资源管理器用来管理 VBA 工程项目。VBE 将每个工作簿作为一个工程，在 Excel 中打开的所有工作簿都集中在工程资源管理器中进行管理。除了工作簿中的工作表以外，还可以管理自定义窗体，以及增加代码模块等，本章将在 3.4 节详细介绍其使用方法。

提示： 如果 VBE 中没有显示工程资源管理器，按快捷键 Ctrl+R 可将其显示出来。

5. 代码窗口

在 Excel 工程中，每一个对象都有一个关联的代码窗口。在【工程资源管理器】中双击对象，即可打开该对象的代码窗口。有关代码窗口的使用方法，将在本章后面进行详细介绍。

6. 工具箱

工具箱只有在设计用户窗体时才会显示出来，使用工具箱中提供的控件可设计出与用户交互的界面。

7. 用户窗体

用户窗体是用户与系统交互的界面，有关用户窗体的设计将在本书第 4 部分进行详细介绍。

Excel 应用程序不使用用户窗体也能完成大部分工作。

8. 属性窗口

属性窗口主要用来设置对象属性。属性窗口显示所选对象的属性，左边为属性名，右边为具体的属性值。属性的设置可直接输入，也可单击下拉列表框进行选择。

属性窗口除了可更改工程、各对象、模块的基本属性外，更多的用于对用户窗体中各对象属性的交互式设计。

提示：如果在 VBE 中没有显示【属性】窗口，按快捷键 F4 可将其显示出来。

9．立即窗口

立即窗口在程序的调试中非常有用，其主要作用是：

- 在开发过程中，可以在代码中加入 Debug.Print 语句，这条语句将在立即窗口输出内容，用来跟踪程序的执行路径，以及变量的中间结果。
- 在调试程序时，如果程序处于中断模式，可以在立即窗口中查看对象和变量的状态。
- 在立即窗口中，使用 Print 语句，就可以看到运行的结果，在很多情况下比用 Msgbox 信息提示对话框方便多了。

提示：如果 VBE 中没有显示【立即窗口】，按快捷键 Ctrl+G 可将其显示出来。

10．本地窗口

本地窗口可自动显示出所有在当前过程中的变量声明及变量值。

若本地窗口为可见的，则每当从执行方式切换到中断模式或是操纵堆栈中的变量时，它就会自动地重建显示。可以通过往左或往右拖移边线，来重置列标头的大小。

3.2　VBE 的子窗口

VBE 操作环境中包含有多个子窗口，例如，工程资源管理窗口、属性窗口、代码窗口、本地窗口、立即窗口等。本节将主要介绍这些子窗口的使用。

3.2.1　工程资源管理窗口

工程资源管理窗口如图 3-5 所示，该图中列出了一个名为“销售管理.xls”的工程。如果在 Excel 中打开了多个工作簿，则在工程资源管理窗口中将列出多个工程名称。

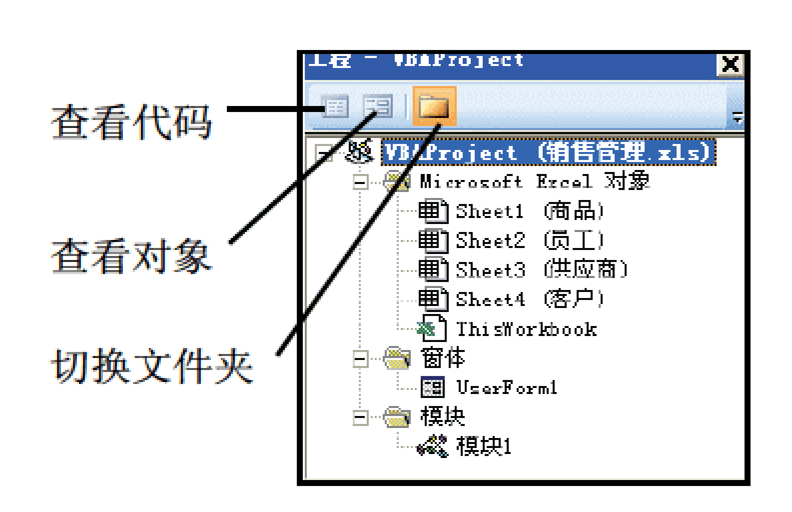


图 3-5　工程资源管理器

在工程资源管理器上方有 3 个按钮，各按钮的作用如下所述：

- ❑ 查看代码：用来显示当前选中模块的代码窗口。
- ❑ 查看对象：用来显示 Excel 对象文件夹中所选择的工作表，或者窗体文件夹里面的窗体。
- ❑ 切换文件夹：用来隐藏或显示工程窗口里的文件夹。

在图 3-5 中展开了工程“销售管理.xls”的各组成部分，可看到其中包括一个【Microsoft Excel 对象】、【窗体】和【模块】节点。在 VBA 工程中可以包括以下对象：

- ❑ 工作表（图表）：包括工作簿中的每个工作表和图表，可对每个对象分别编写代码。
- ❑ 模块：包括在 Excel 中录制的宏，以及用户编写的 VBA 代码。
- ❑ 类模块：包括用户自定义对象的代码。
- ❑ 窗体：包括该工程中设计的用户自定义窗体，以及为窗体、窗体控件编写的代码。

3.2.2 属性窗口

属性窗口用来查看或设置对象的属性，可设置工程、工作表、窗口、模块、类模块等对象的属性值，在大多数情况下，主要用来设置用户窗体及窗体中使用的控件属性。

常见的属性窗口如图 3-6 所示。当前所选对象的名称显示在属性窗口的【对象】列表框中，单击右侧的下拉箭头，可查看对象的名称列表，在该列表框中选择某个对象名后，也就选中了该对象。

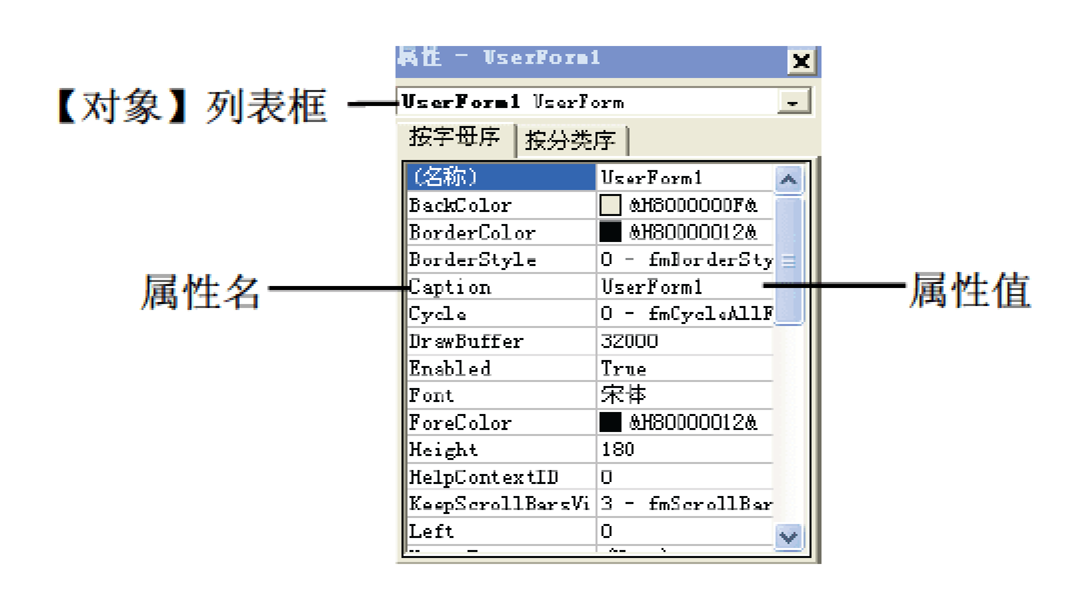


图 3- 6 【属性】窗口

在【属性】窗口中，最大的区域就是设置对象各属性列的列表，该列表分两列，左侧为属性名（用户不能修改），右侧为属性可设置的值。

设置对象属性时，根据属性的不同，需要使用不同的设置方法。有的属性值需要手工输入；有的是在列表中进行选择；还有的需要打开一个对话框进行选择。

1. 手工输入属性值

大部分属性值都是通过手工输入，例如对象的名称、外形尺寸等，这类属性值的设置很简单。例如，要将新插入的用户窗体名称设置为 frmMain，具体操作步骤如下：

（1）单击主菜单【插入】|【用户窗体】命令，向工程中插入一个用户窗体。

（2）在【工程】窗口中双击新插入的用户窗体，这时【属性】窗口将显示用户窗体的属性，拖动鼠标选中属性【（名称）】右侧的文字 UserForm1，如图 3-7 所示。

（3）接着输入新的名称 frmMain，按 Enter 键后【属性】窗口的名称已经修改，如图 3-8 所示。

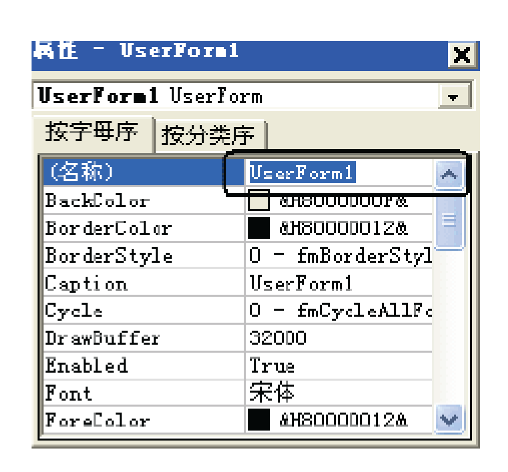


图 3-7　选择属性值



图 3-8　修改属性

2. 选择列表

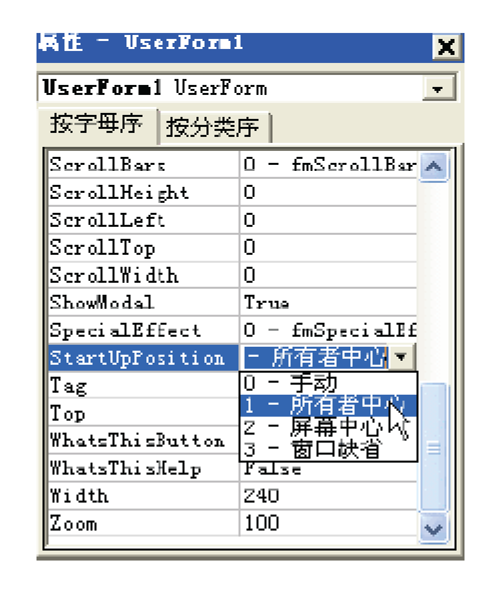
有的属性取值具有一个规定的范围，在属性窗口中将提供一个列表供用户进行选择。例如，设置用户窗体的位置属性的操作步骤如下：

（1）在属性窗口中找到需要设置的属性名 StartUpPosition；

（2）单击属性值右侧的下拉箭头▼，将显示出可供选择的列表，如图 3-9 左图所示；

（3）单击列表中需要设置的值“1 – 所有者中心”。

另外，在如图 3-9 右图所示列表框中可设置窗体的背景色。



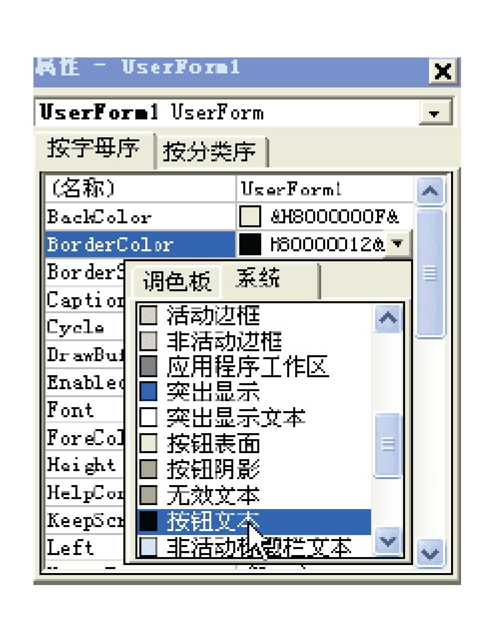


图 3-9　选择属性值

3. 打开对话框

对于设置属性为一个文件名之类的值时，需要打开一个对话框，让用户选择相应的文件。例如，设置窗体背景图片的操作步骤如下：

（1）在属性窗口中找到需要设置的属性名 Picture。

（2）单击属性值右侧的按钮...，将打开如图 3-10 所示的【加载图片】对话框。

（3）在该对话框中选择合适的图片文件后，单击【打开】按钮，【属性】窗口如图 3-11 所示，在 Picture 属性后显示为（Bitmap），表示为该属性设置了一个图片文件。同时，窗体背景将显示该图片。

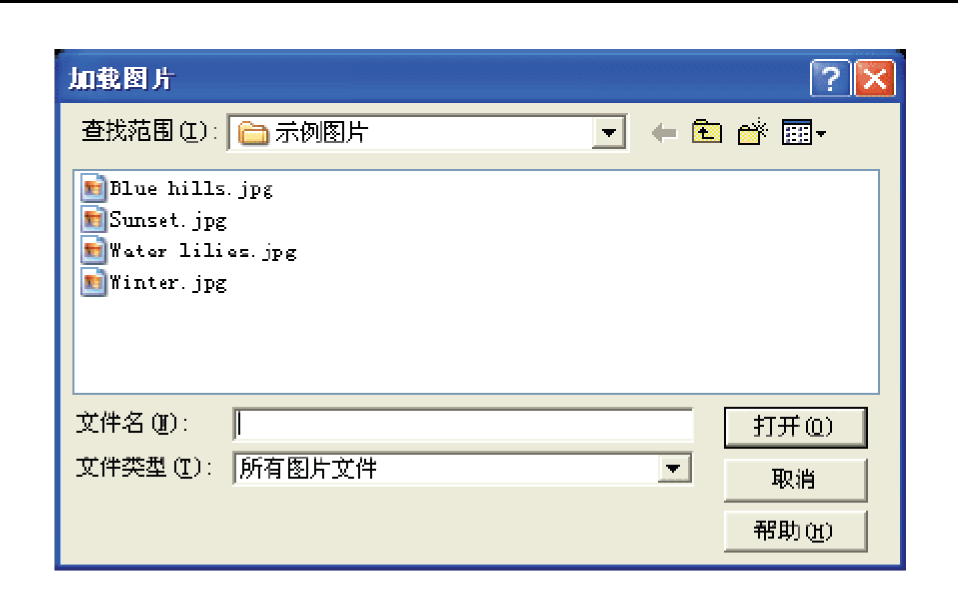



图 3-10 【加载图片】对话框

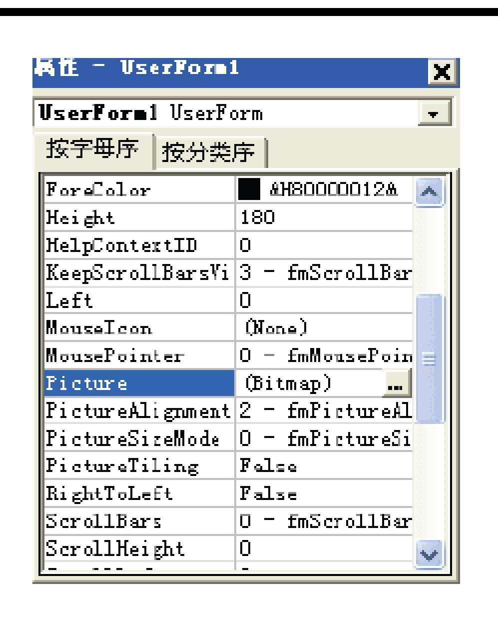


图 3-11 设置 Picture 属性

3.2.3 代码窗口

在 VBE 中，每个模块都有自己的代码窗体，每个工作表、每个模块、每个窗体都分别有自己的代码窗口。代码窗口如图 3-12 所示。

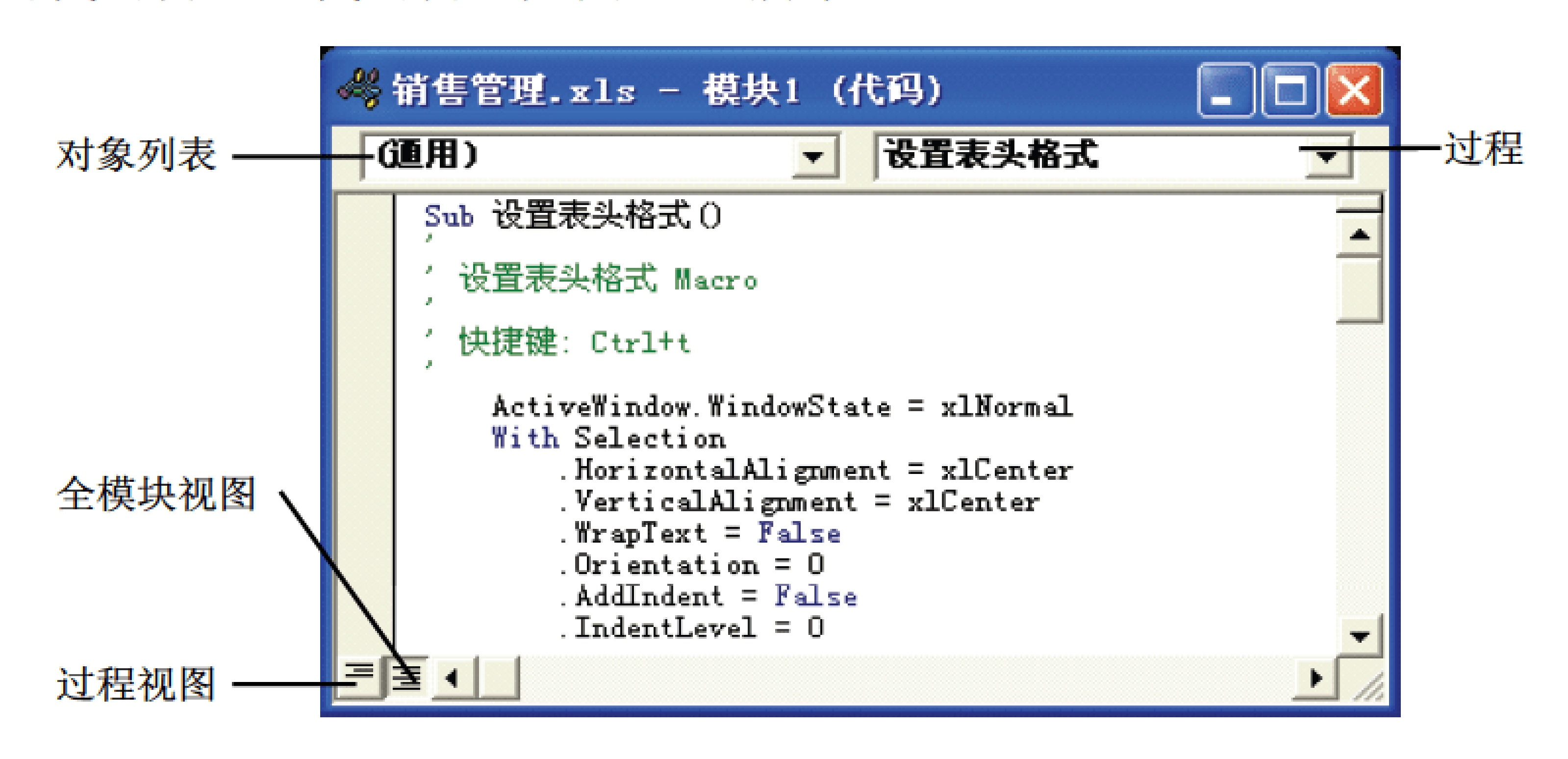


图 3-12 代码窗口

在代码窗口的顶部，有两个下拉列表框。左侧为【对象列表】下拉列表框，在该列表框中将显示当前模块的对象名列表。右侧为【过程/事件】下拉列表框，可快速查看一个指定对象过程或事件过程的代码。

在代码窗口的左下方有两个按钮，其功能包括以下两方面。

- 过程视图：在代码窗口中一次只显示一个过程的代码。如果要查看其他过程的代码，则需要通过【过程/事件】下拉列表框选择过程。
- 全模块视图：在代码窗口中显示所选模块的所有过程，使用右侧的垂直滚动条可以在代码中滚动。

3.2.4 调整 VBE 子窗口位置

VBE 环境由多个子窗口组成，这些子窗口可根据需要显示或隐藏，各子窗口可以停靠在主窗口的边上，也可浮动显示在窗口中，可根据用户的习惯进行定制。下面演示在 VBE

中拖动各子窗口的操作。

（1）在 Excel 2007 打开一个工作簿，或新建一个工作簿。

（2）按快捷键 Alt+F11 进入 VBE 开发环境，如图 3-13 所示。

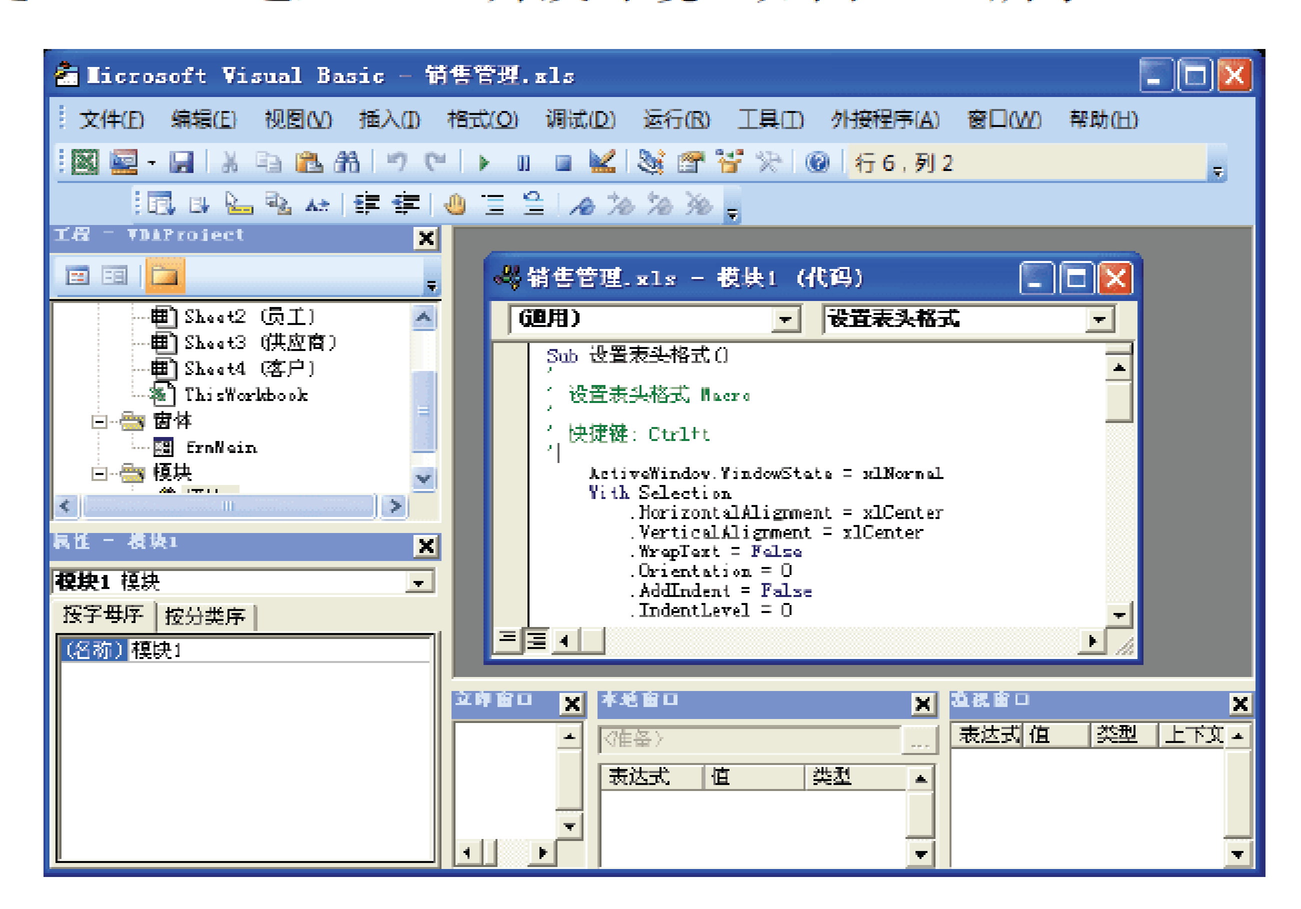


图 3-13　VBE 开发环境

（3）单击各子窗口右上方的【关闭】按钮，可隐藏子窗口，如图 3-14 所示为关闭所有子窗体后的效果。

（4）单击主菜单【视图】|【工程资源管理器】命令（如图 3-15 所示），可显示出【工程资源管理器】子窗口。

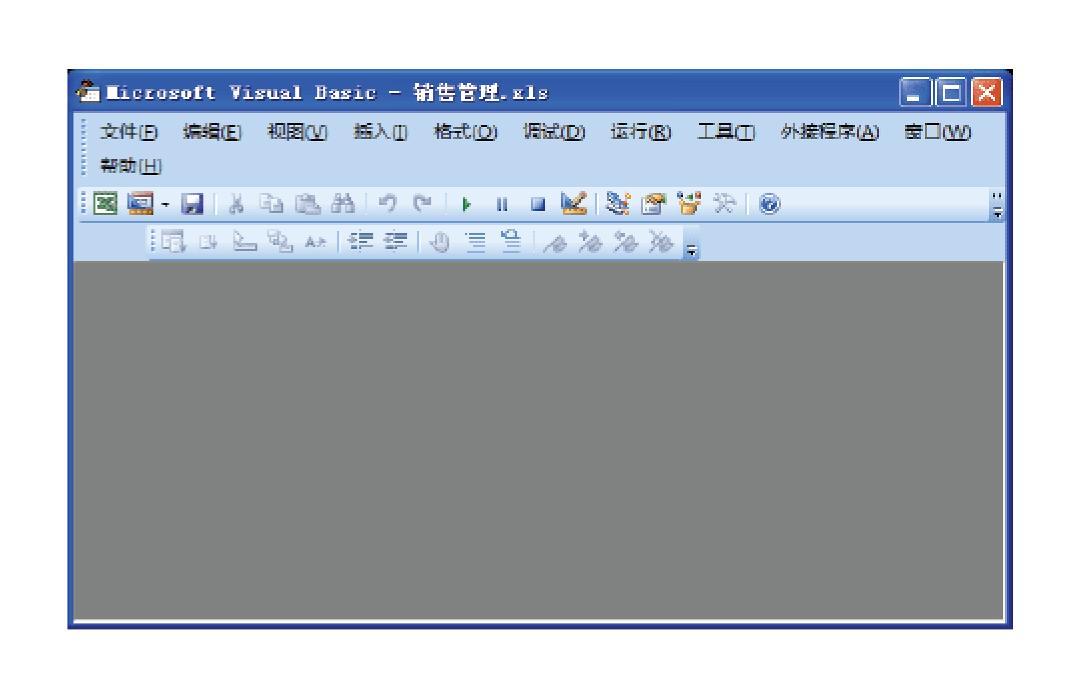


图 3-14　关闭各子窗体后的效果

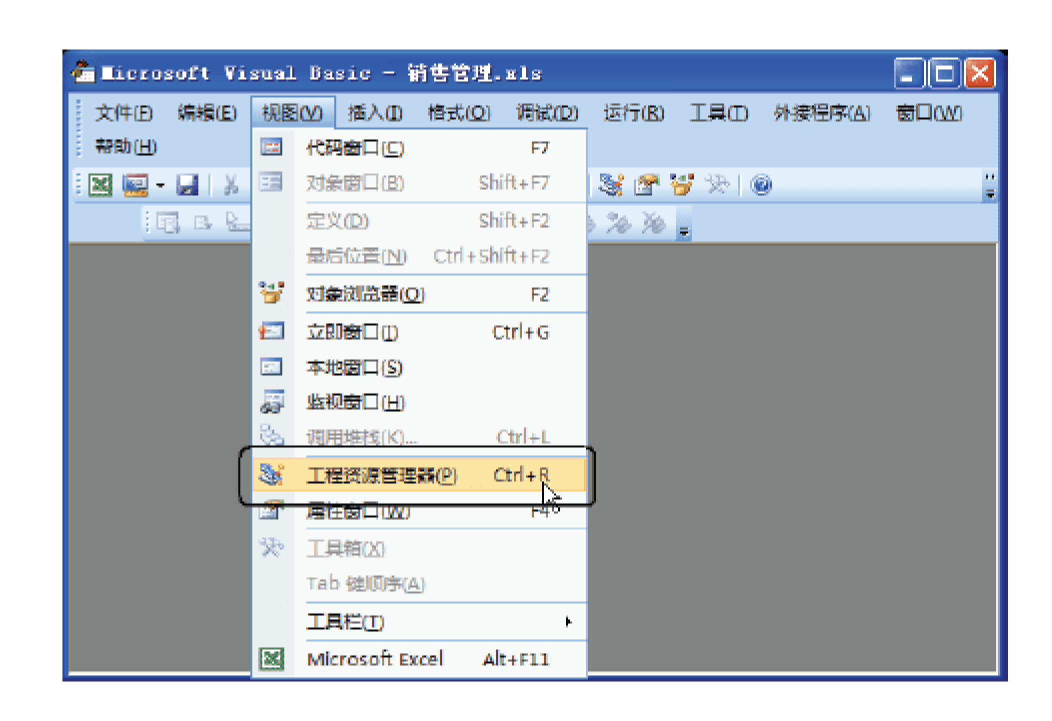


图 3-15　【视图】下拉菜单

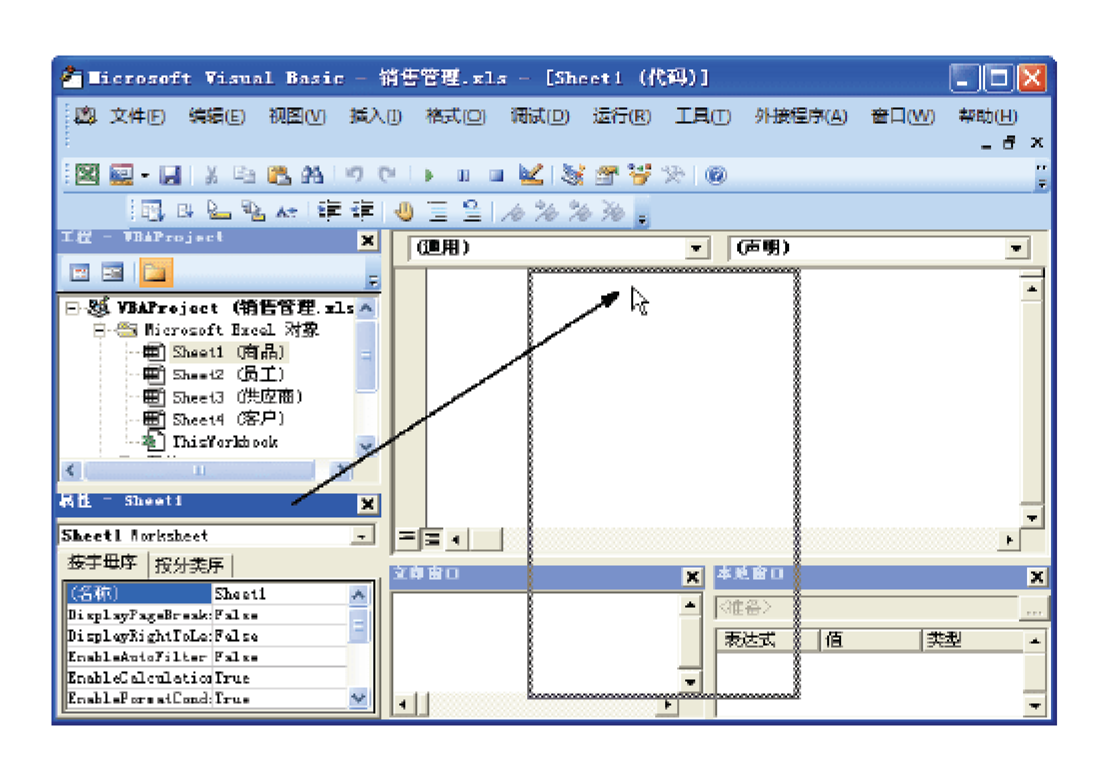
（5）在【视图】下拉菜单中选择不同的命令，将常用子窗口显示出来。

（6）拖动【属性】子窗口的标题栏向右移动，使【属性】子窗口处于浮动状态，如图 3-16 所示。

（7）右击【属性】子窗口标签右侧的空白处，弹出如图 3-17 所示的快捷菜单，默认情况下【可连接的】菜单项处于选中状态，此时该子窗口靠近主窗口或其他子窗口时，将自动连接到靠近的窗口。

（8）单击【可连接的】菜单项，取消其选中状态。此时，拖动【属性】子窗口调整其位置，该子窗口将不会连接到靠近的子窗口旁，而会隐藏在其他子窗口下方，如图 3-18

所示。



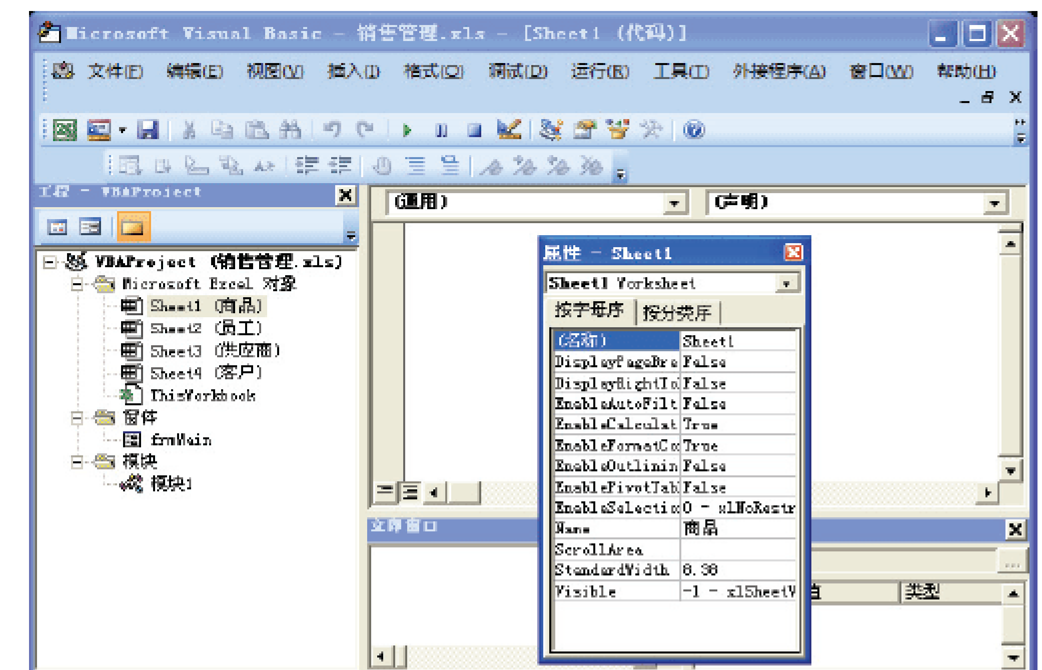
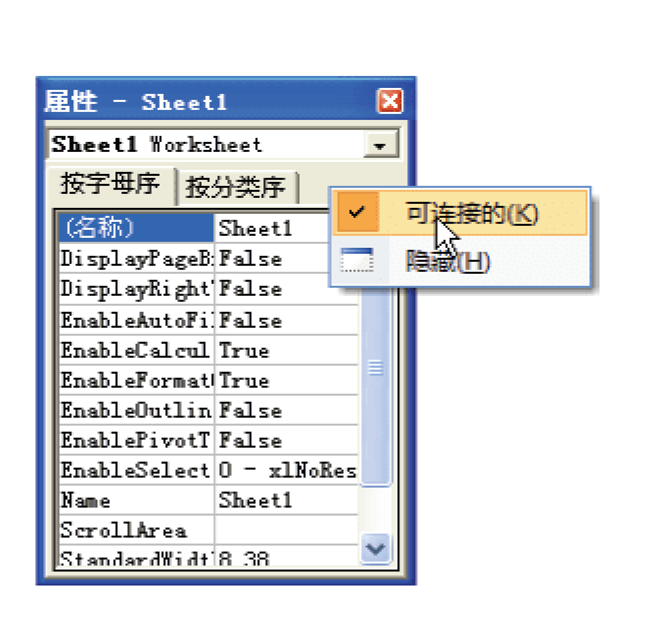


图 3-16 拖动【属性】子窗口



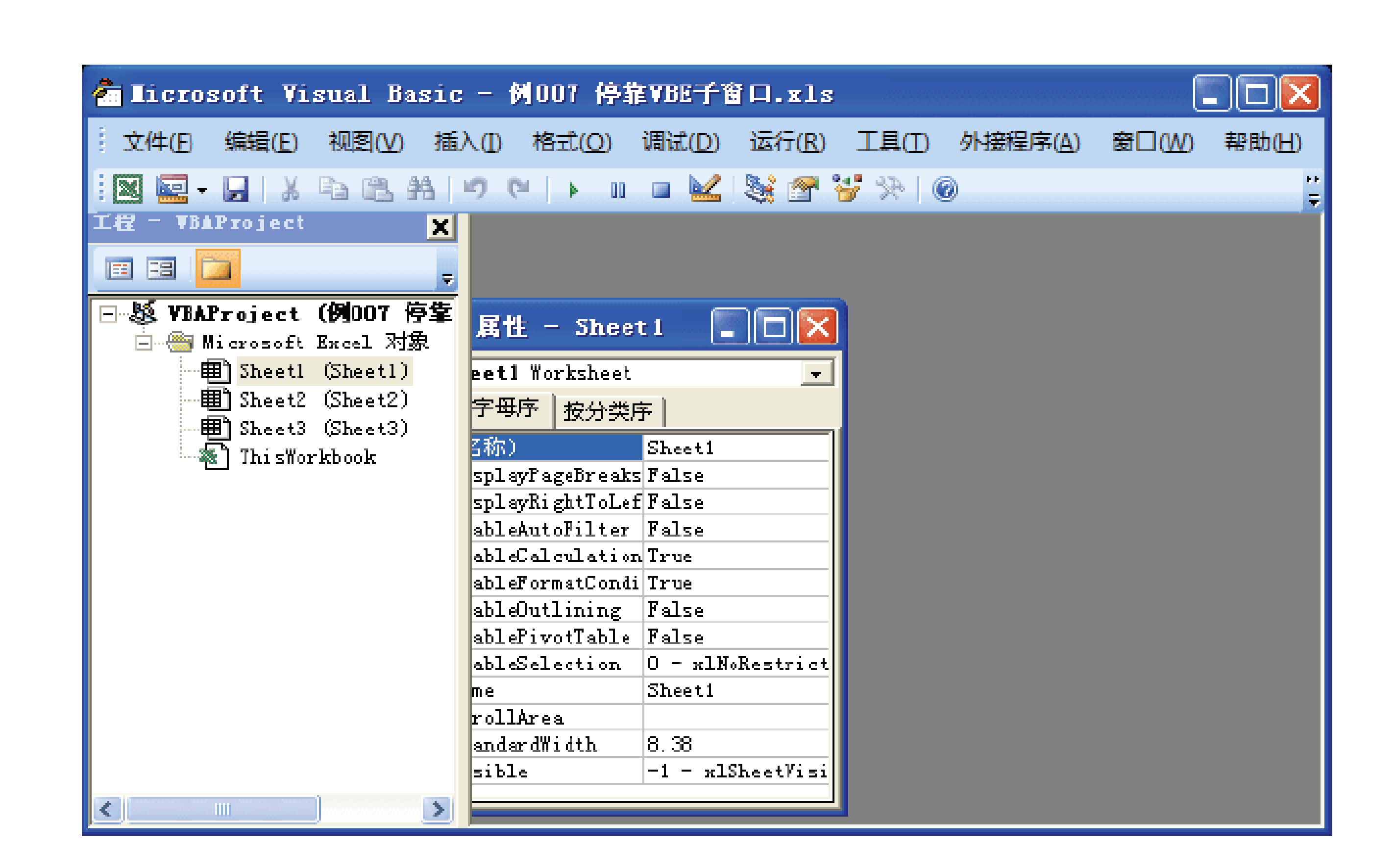


图 3-17 【属性】子窗口的快捷菜单

图 3-18 隐藏部分【属性】子窗口

3.3 定制 VBE 环境

通过上面的学习，读者已经对 VBE 环境有了较全面的认识。对于 Excel 开发人员来说，还应当根据自己的习惯，对 VBE 环境进行定制。

在 VBE 环境中单击主菜单【工具】|【选项】命令，打开【选项】对话框。该对话框包括 4 个选项卡，通过这些选项卡中的选项可对 VBE 环境进行定制。

3.3.1 设置【编辑器】选项卡

【选项】对话框中的【编辑器】选项卡如图 3-19 所示。通过该选项卡中的选项可定制【代码】窗口。下面分别介绍这些选项。

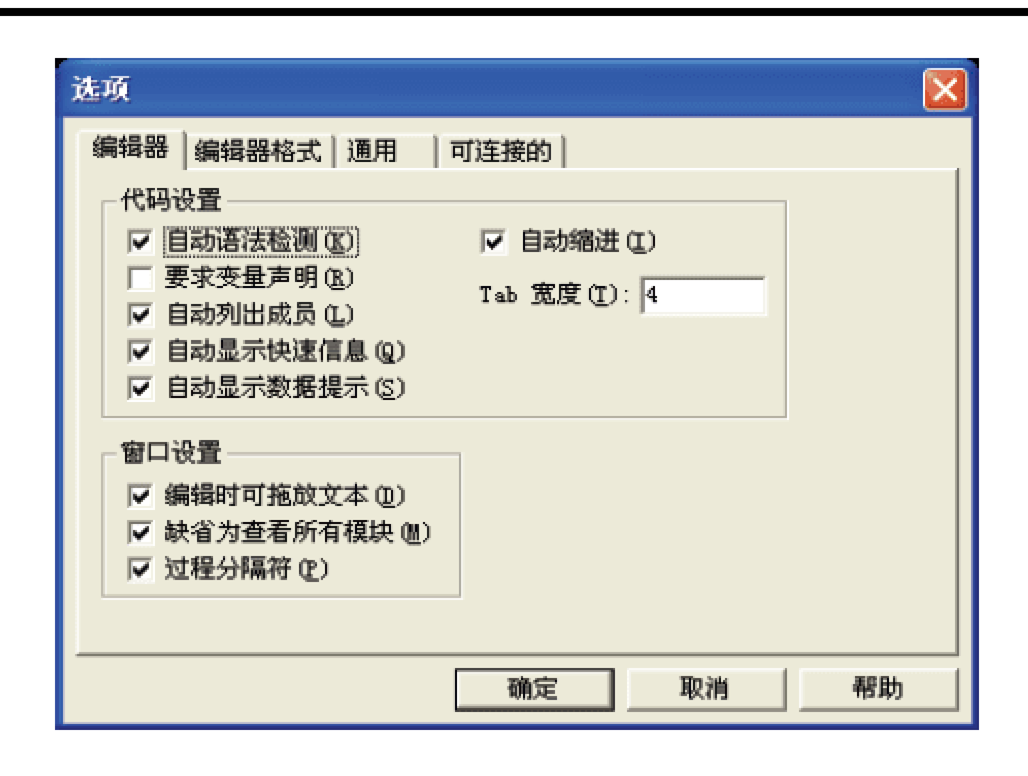


图 3-19 【编辑器】选项卡

- 自动语法检测：在输入一行代码之后，是否自动检查语法。如果未选中该复选框，VBE 将通过使用与其他代码不同的颜色来显示语法错误的代码，并且不弹出提示对话框。
- 要求变量声明：选中该选项，VBE 将会在新插入的模块起始处增加以下语句。

```
Option Explicit
```

如果该语句出现在模块中，就必须定义模块中使用的每个变量。当遇到未定义的变量时，将出现错误提示。

- 自动列出成员：选中该项，在输入 VBA 代码时，VBE 将自动列出对象的成员列表。如果没有选中该项，则需要单击【编辑】工具栏中的【属性/方法列表】按钮（或按 Ctrl+J 快捷键）来打开该成员列表。
- 自动显示快速信息：选中该项将显示所输入函数及其参数的信息。
- 自动显示数据提示：选中该项将显示出指针所在位置的变量值。只能在中断模式下使用。
- 自动缩进：选中该项，VBE 将按照下方设置的 Tab 宽度自动缩进显示每行代码，所有接下来的代码都将使用该缩进量。
- Tab 宽度：设置缩进量，范围为 1～32 个空格，默认值是 4 个空格。
- 编辑时可拖放文本：选中该项后，允许通过拖动操作来复制和移动代码窗口中的文本。也可从代码窗口拖放元素到立即窗口或监视窗口。
- 缺省为查看所有模块：设置新模块的默认状态，选中该项后，在代码窗口中可查看模块的所有过程。这不会改变当前已打开模块的视图方式。
- 过程分隔符：可以显示或隐藏代码窗口中，出现在每个过程尾端的分隔符条。

3.3.2　设置【编辑器格式】选项卡

通过【选项】对话框的【编辑器格式】选项卡，可设置代码的显示格式。【编辑器格式】选项卡如图 3-20 所示，可设置各种代码的显示颜色、字体大小等内容。具体设置步骤

如下：

（1）在【代码颜色】列表框中选择需要调整的代码类型，如语法错误文本。

（2）在【前景色】、【背景色】和【标识色】的下拉列表框中选择“自动”选项。

（3）在右侧的【字体】和【大小】列表框中分别设置代码的字体和字号。

（4）根据需要，还可选择是否显示【边界标识条】。

在设置过程中，可通过【示例】文本框观察到设置的效果。

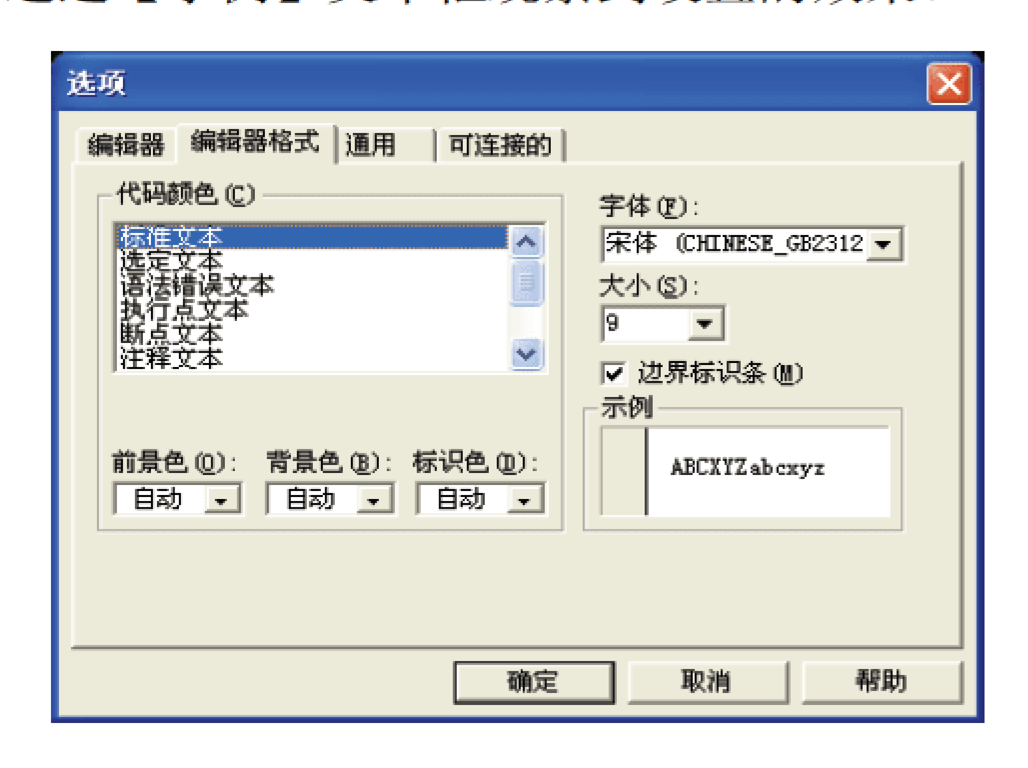


图 3-20　【编辑器格式】选项卡

3.3.3　设置【通用】选项卡

【选项】对话框的【通用】选项卡如图 3-21 所示，主要用来进行 VBE 的工程设置、错误处理和编译设置。一般情况下，不需要修改这些选项，直接使用默认值即可。

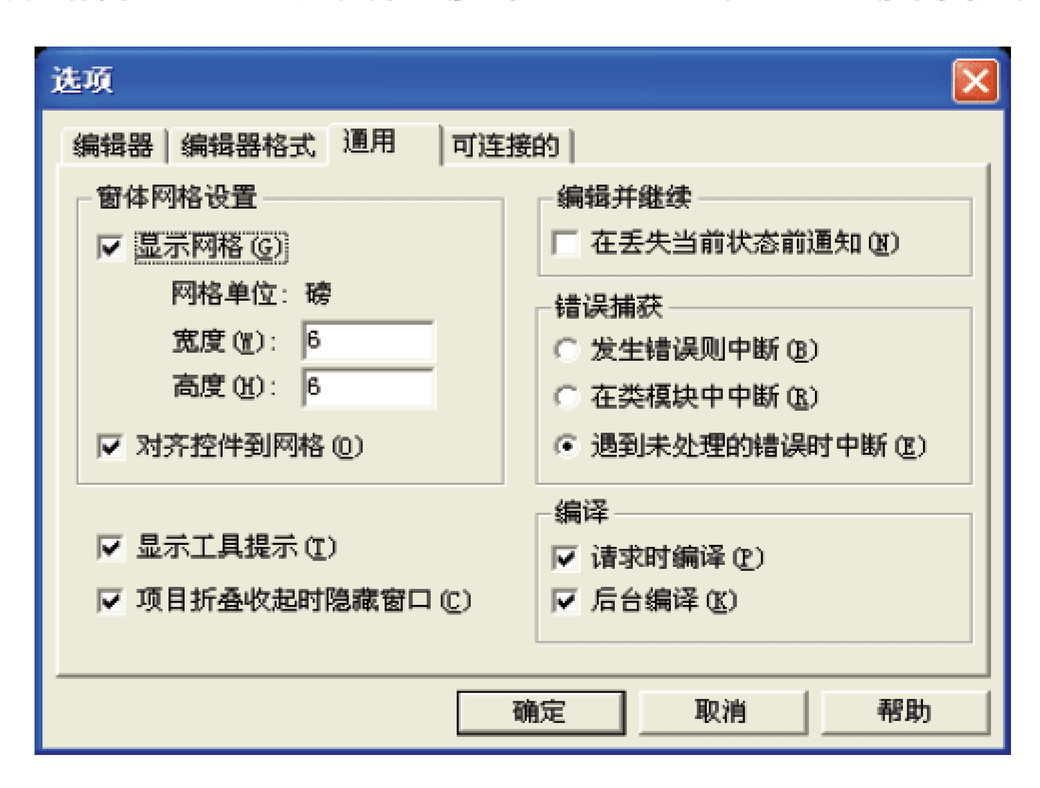


图 3-21　【通用】选项卡

【通用】选项卡中各选项的意义如下所述。

- ❑ 显示网格：在窗体中显示网格。
- ❑ 网格单位：列出网格中单元的度量单位。
- ❑ 宽度：窗体中网格存储单位的宽度。
- ❑ 高度：窗体中网格存储单位的高度。

- 对齐控件到网格：在封闭网格上自动对超过控件边缘的部分定位。
- 显示工具提示：显示工具栏按钮的工具提示。
- 项目折叠收起时隐藏窗口：当【工程】资源管理器窗口中的工程折叠起来之后，会自动关闭工程、用户窗体、对象或模块窗口。
- 在丢失当前状态前通知：显示一个信息，指出对运行中的工程所要求的动作，以使所有模块级变量得以重置。
- 发生错误则中断：任何错误都会使工程切换到中断模式，不管错误处理程序是否为活动的，也不管代码是否在类模块中。
- 在类模块中中断：任何在对象类模块中失去句柄的错误会让工程进入中断模式。
- 遇到未处理的错误时中断：任何其他失去句柄的错误会让工程进入中断模式。
- 请求时编译：在执行前就已完全编译工程，或已编译所需的代码。
- 后台编译：利用运行时的空闲时间在后台完成工程的编译（只有在已设置请求时编译才有效）。

3.3.4　设置【可连接的】选项卡

【选项】对话框的【可连接的】选项卡如图 3-22 所示，其中的选项决定 VBE 中各窗口的行为方式。

可连接可理解为可停靠，如在默认情况下，将【工程】资源管理器窗口拖到 VBE 窗口的边缘时，【工程】资源管理器窗口将自动停靠在 VBE 窗口边框上。如果在图 3-22 所示的对话框中没有选中对应窗口前的复选框，则该窗口将浮动在 VBE 窗口上，当有多个窗口浮动时，VBE 操作界面将显示得很混乱。

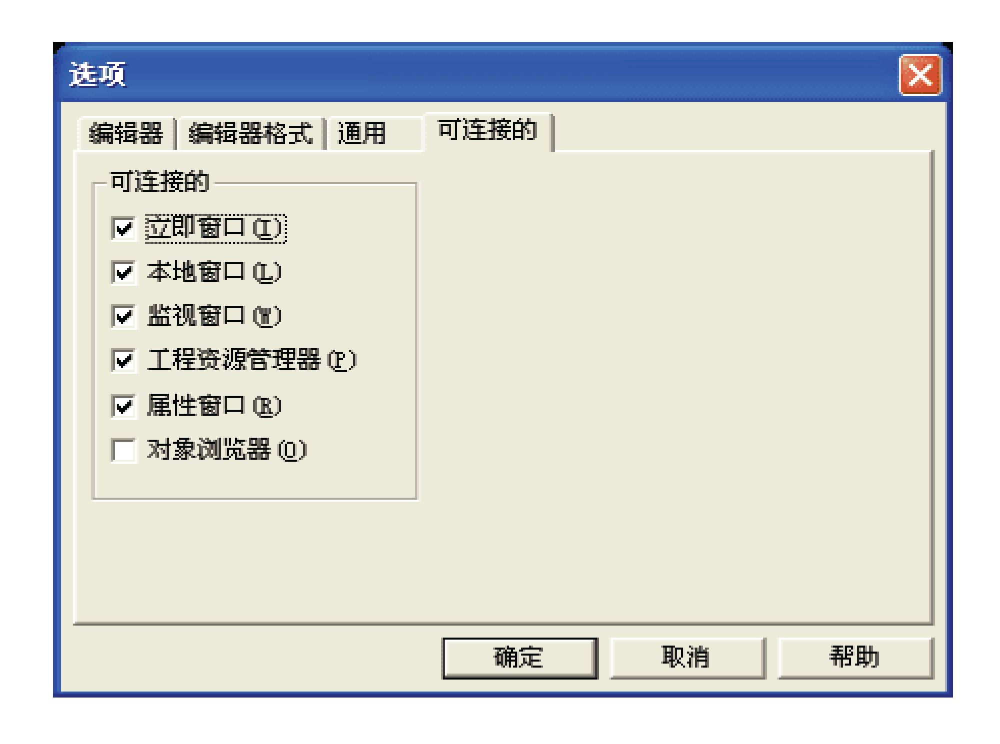


图 3-22　【可连接的】选项卡

3.4　使 用 帮 助

VBE 为开发人员提供了完善的帮助系统，可帮助用户学习 VBA 相关对象的属性、方法、事件的全面信息。本节简单介绍帮助系统的使用方法。

3.4.1 打开帮助主界面

在 VBE 环境中单击主菜单【帮助】|【Microsoft Visual Basic 帮助】命令，可打开如图 3-23 所示的帮助主界面。



图 3-23　帮助主界面

提示：按 F1 键也可打开如图 3-23 所示的帮助主界面。

在帮助主界面中，主要包括以下几个部分。

- 工具栏：类似于 IE 浏览器的工具栏，工具栏各按钮如图 3-24 所示。
- 搜索栏：可输入关键字，在帮助系统中查找与关键字匹配的帮助信息。
- 目录：显示 Excel 2007 中的所有帮助信息目录。
- 主窗口：显示帮助的具体内容。

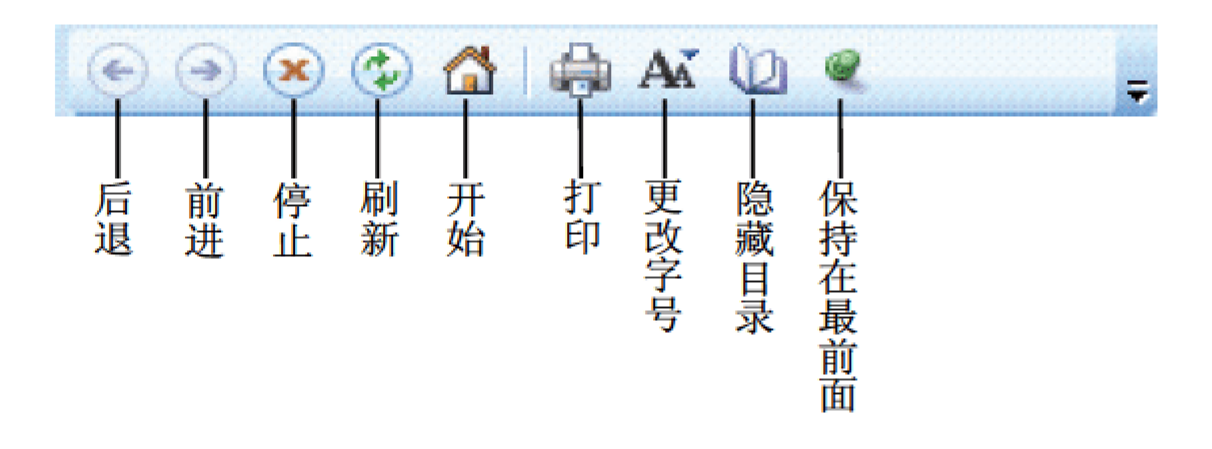


图 3-24　工具栏的各按钮

3.4.2 查看对象属性

要查看对象的属性，可在帮助窗口左侧的目录中依次单击展开对象名称，找到对象的属性列表即可查看到该属性的相关说明，如图 3-25 所示。

提示：在帮助正文中，有的关键字可链接到其他帮助条目（如图 3-25 中的 Range 关键字），单击该链接可打开对应的帮助条目。

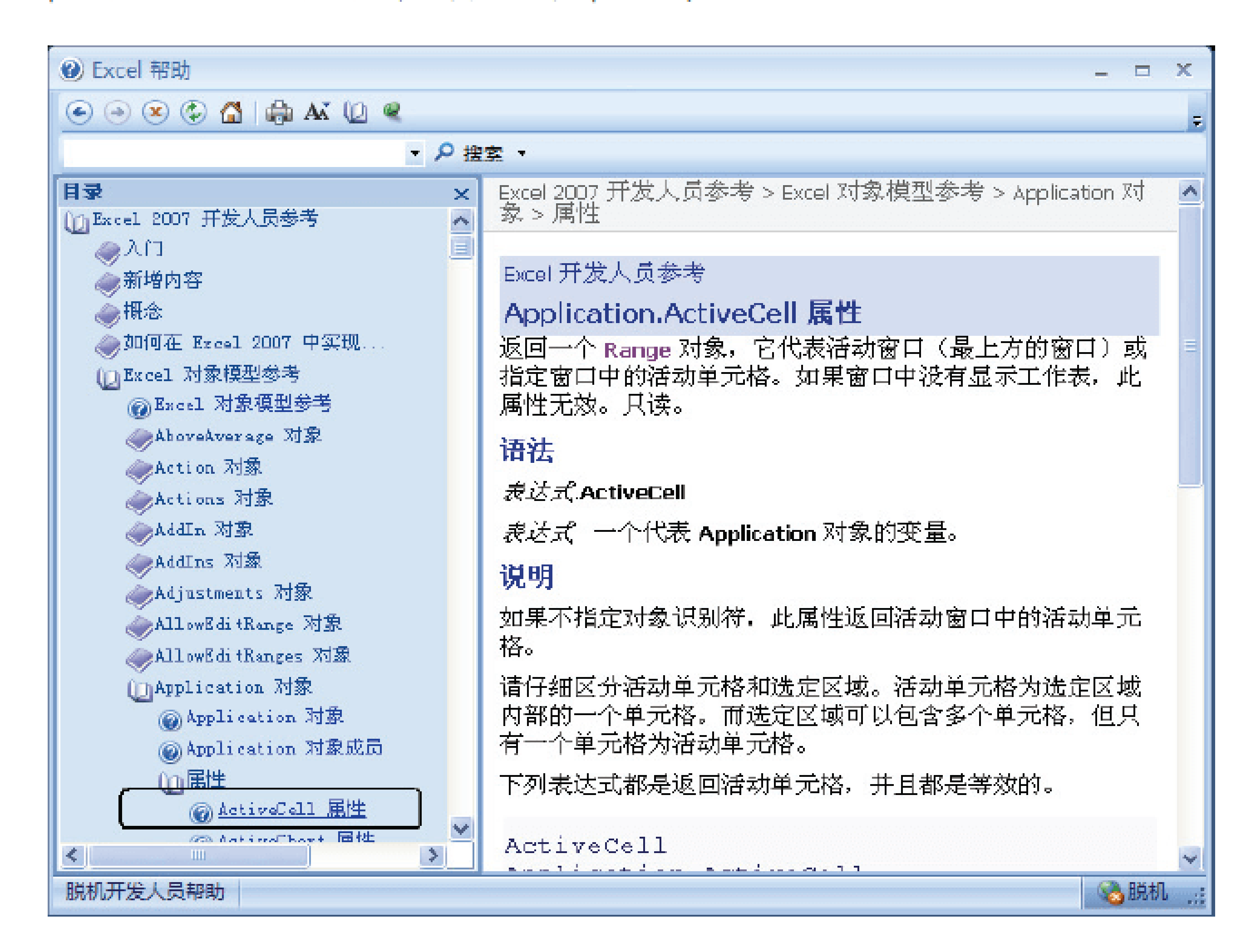


图 3-25　查看对象的属性

3.4.3　搜索关键字

在帮助系统中，用户可输入关键字（或关键字的部分字母），单击【搜索】按钮，即可在帮助系统中搜索包含该关键字的相关内容，如图 3-26 所示。

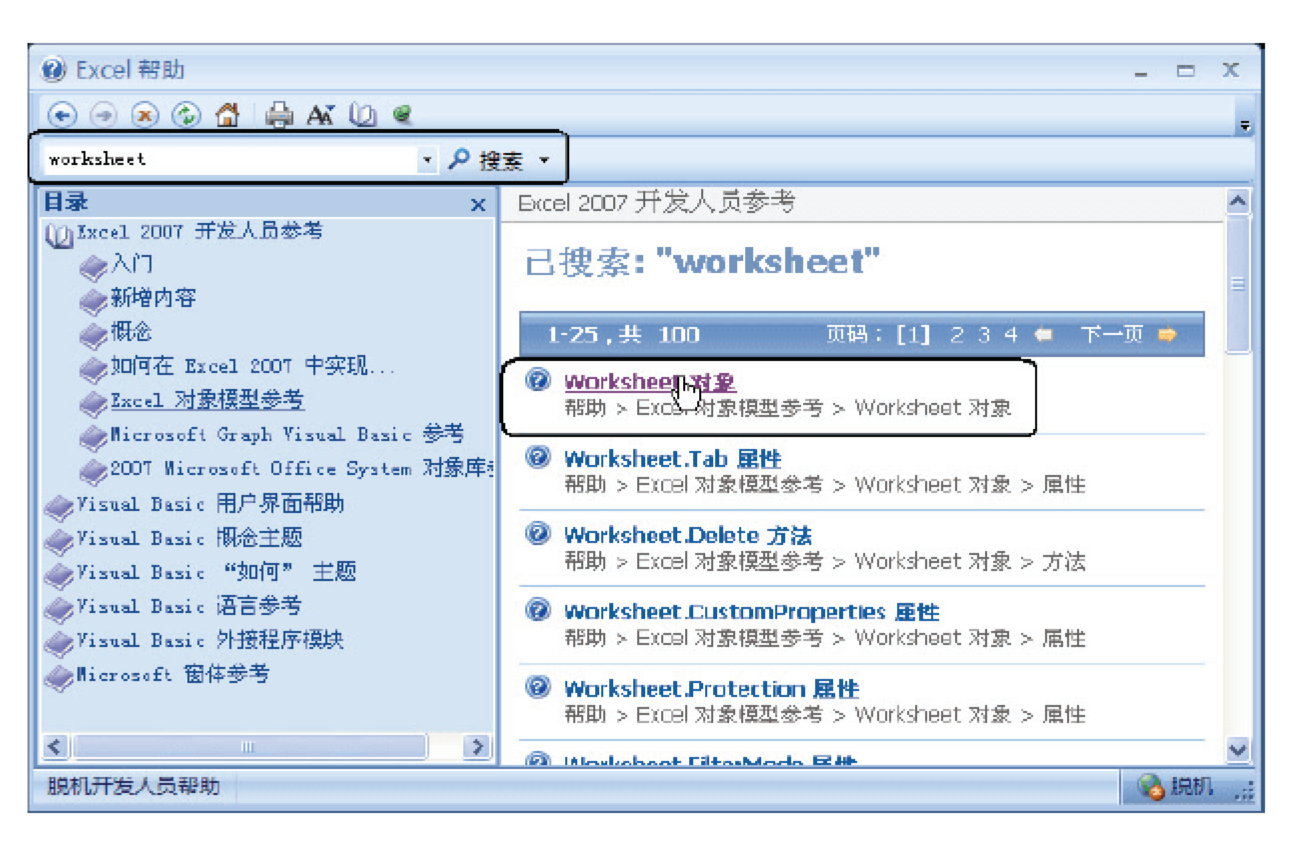


图 3-26　搜索关键字

将鼠标移到要查看的条目上，鼠标指针将变为手形，单击鼠标即可打开相应条目的帮助信息。

如果帮助系统中与搜索关键字相匹配的条目很多，将显示分页序号，单击序号链接可查看其他页的内容。

第 2 部分　VBA 基础知识

在使用 VBA 编写代码前，必须先了解 VBA 程序设计的基础知识。了解 VBA 程序的基本元素包括：语句的书写规范，VBA 处理的数据类型，程序的结构、字符串和日期的处理等相关知识。

第 4 章　VBA 基础

在用 VBA 开发应用程序之前，应该先对 VBA 的语法有个了解。本章将介绍 VBA 数据类型、常数、变量、运算符和表达式等相关基础知识，为进一步学习 VBA 编程打下基础。

4.1　VBA 简介

在 Excel 中使用 VBA 编程，可以开发出许多有价值的应用程序。作为初学者，首先需要了解什么是 VBA，以及用 VBA 能做什么。

4.1.1　什么是 VBA

VBA 是 Visual Basic for Application 的缩写，是一种应用程序自动化语言。所谓应用程序自动化，是指通过编写程序让常规应用程序（如 Excel、Word 等）自动完成工作，如在 Excel 里自动设置单元格的格式、多张工作表之间的自动计算等。

VBA 是微软应用程序开发语言——VB 的子集。所以如果有 VB 程序设计的经验，学习 VBA 将非常快；同样，如果掌握了 VBA 的应用，也会为以后学习 VB 打下坚实的基础。

VBA 根据其嵌入软件的不同，增加了对相应软件中对象的控制功能。例如 Excel 的 VBA，增加了控制 Excel 工作簿、工作表、区域、数据透视表等对象的属性、事件和方法。在 Excel 中使用 VBA，可以更好地控制 Excel，进一步发掘 Excel 的强大功能，提高 Excel 的自动化水平。可以用很短的时间在 Excel 环境中开发出一套完整的管理信息系统。

4.1.2　在 Excel 中使用 VBA 的优势

以 Excel 为平台，用 VBA 来开发程序的优势有以下几点：

- 当使用 Excel 为平台时，利用 Excel 现有的功能（如文件管理、函数等），可以减少应用程序的代码量从而大大缩短开发的周期。
- 在大部分用户的计算机中都安装有 Excel 软件，使 Excel 开发的应用程序发布很容易。只要用户计算机中有 Excel，基本不需要其他的文件，只需将开发的工作簿文件复制给用户即可完成文件的发布。
- VBA 的语言简单易学，初学者很容易上手。

4.2　VBA 语法简介

在 Excel VBA 中，每个 Excel 应用程序都由一个工程表示，每个工程包含当前工作簿的工作表、用户窗体、模块和类模块，可为这些对象和模块分别编写 VBA 代码，本节介绍 VBA 代码最基础的内容。

4.2.1　了解 VBA 代码

在本书第 2 章中录制的宏都是由 VBA 代码构成的，进入 VBE 环境可以看到录制的宏代码。在 Excel 中录制的宏都是以 VBA 代码表示，所有的宏都是以关键词 Sub 开始，以关键词 End Sub 结束。其结构如下：

```
Sub 宏名称()
' 说明

   VBA 语句 1
   VBA 语句 2
   ……
End Sub
```

在关键词 Sub 之后是宏的名称，宏名称后是一对括号。在 Sub 和 End Sub 之间是每次执行宏时的 VBA 语句，其中以单引号（半角）开头的是注释内容，VBA 将忽略该语句。

提示：以 Sub 开始，至 End Sub 结束的这一部分代码在 VBA 中称为一个过程。有关 Sub 过程和 Function 过程的详细内容将在本书第 7 章中进行介绍。

例如，第 2 章中录制宏的 VBA 代码如下：

```
Sub 设置表头格式()
' 设置表头格式 Macro
' 快捷键: Ctrl+t
    ActiveWindow.WindowState = xlNormal
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlCenter
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    Selection.Merge
    With Selection.Font
```

```
        .Name = "宋体"
        .Size = 20
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ColorIndex = xlAutomatic
        .TintAndShade = 0
        .ThemeFont = xlThemeFontNone
    End With
    Selection.Font.Bold = True
    ActiveWindow.WindowState = xlNormal
End Sub
```

在以上代码中可以看出，VBA 代码主要有以下几个要素：

- ❑ VBA 代码由英文、中文、数字等字符构成。
- ❑ 每行为一个 VBA 语句。
- ❑ 有一些固定的单词（称为关键字）经常出现，例如，Selection、With 等。
- ❑ 几乎每行代码中都包括句点，用来连接 VBA 语言中不同的要素。例如，ActiveWindow.WindowState。

下面将分别介绍这些语法要素。

4.2.2 VBA 字符集

VBA 代码可由以下几类字符组成。

- ❑ 英文大小写字母：包括 A～Z 和 a～z。VBA 的代码不区分大小写，若字母作为字符串使用时，需注意大小写不同。
- ❑ 数字：包括阿拉伯数字 0～9。
- ❑ 特殊符号：在 VBA 中具有特殊含义的字符，如+、–、*、/、>、<、=和各种标点符号。
- ❑ 汉字：对于中文版的 Excel，在 VBA 代码中还可以使用汉字。

综上所述，在 VBA 代码中可使用各种常见的符号。

4.2.3 关键字

关键字，又称为保留字，是指 VBA 中具有特殊意义的保留字或符号。这些关键字具有固定的含义，用户不能更改其拼写方式，在对过程、变量命名时不允许使用关键字相同的拼写。

VBA 中关键字很多，可分为数组、编译命令、控制流、变换、数据类型、日期与时间、目录和文件、错误、金融、输入与输出、数学、其他、操作符、字符串处理、变量与常数等多种类型。常用的关键字如表 4-1 所示。

表 4-1　VBA常用关键字

Abs	Array	Asc	Atn	Boolean	Byte	Call	Case	Choose	Chr
Close	Compare	Const	Currency	Date	Day	Deftype	Dim	Dir	Do
Double	Else	End	Exit	Explicit	For	Format	Function	Hour	If
InStr	Int	Integer	Is	Lbound	Lcase	Len	Let	Like	Long
Loop	Lset	Ltrim	Me	Mid	Minute	Mod	Month	New	Next
Not	Now	On	Open	Option	Or	Print	Private	Public	QBColor
Randomize	ReDim	Return	Right	RmDir	Rtrim	Second	Select	Set	Sgn
Sin	Single	Space	Spc	Static	String	Sub	Switch	Tab	Then
Timer	Trim	Ucase	Wend	While	With	Xor	Year		

提示：在 VBA 中，对关键字不区分大小写，无论用户是按大写、小写或大小写混合的方式输入关键字，当输入完一条语句按 Enter 键后，VBE 将自动将关键字转换为首字母大写，其余字符小写的样式。

4.2.4　标识符

在 VBA 程序中，为了区分过程、常数、变量、对象等，需要为这些过程、常数、变量、对象分别设置不同的名称，这个名称就是标识符。

用户定义标识符时应按照以下规则来定义：

- 标识符只能由 VBA 支持的字符集组成。
- 标识符首字母必须为字母或下划线。
- 不能在标识符中使用空格、句点（.）、感叹号（!）或@、&、$，# 等字符。
- 标识符不能与关键字相同，可对关键字加上前缀或后缀使用。
- 标识符的长度不能超过 255 个字符。
- 标识符的名称应尽量有意义，以方便程序中查错。
- 中文 Excel 中，可使用中文作为标识符（这时，不要求首字符为字母），如设置过程名称为“设置表头格式”。

4.3　数 据 类 型

数据是程序的处理对象，在学习程序设计之前，有必要先了解数据的相关知识。VBA 能处理字符、日期、数值等多种数据类型，并允许用户根据需要定义自己的数据类型。

4.3.1　基本数据类型

Excel 单元格中可以保存处理多种类型的数据，包括数值、日期/时间、文本、货币等。

在 VBA 中除了提供这些数据类型之外，还提供了字节、布尔和变体数据等类型。在 VBA 中共有 11 种基本数据类型，这些基本数据类型是组成用户自定义类型的基础。

1. 整型（Integer）

整型数据存储为 16 位（2 个字节）的数值形式，其范围为–32768～32767 之间。整型数据除了表示一般的整数外，还可以表示数组变量的下标。整型数据的运算速度较快，而且比其他数据类型占用的内存少。整型的类型声明字符是百分比符号（%），以下两条语句都声明了一个整型变量。

```
Dim n1 As Integer
Dim n%
```

2. 长整型（Long）

长整型数据存储为 32 位（4 个字节）有符号的数值形式，其范围从–2147483648～2147483647。Long 的类型声明字符为和号（&）。

如果要在 VBA 程序中保存较大数值，可采用长整型来保存。

3. 单精度浮点型（Single）

整型和长整型都用来表示整数，在很多程序中都需要处理小数，这时就需要使用实数型，实数型分为单精度浮点型和双精度浮点型。单精度浮点型数据存储为 32 位（4 个字节）浮点数值的形式，通常以指数形式（科学计数法）来表示，以“E”或“e”表示指数部分。它的范围在负数的时候是从–3.402823E38～–1.401298E-45，而在正数的时候是从 1.401298E-45～3.402823E38。单精度浮点型的类型声明字符为感叹号（!）。

4. 双精度浮点型（Double）

双精度浮点型可表示更高精度、更大的数据，存储为 64 位（8 个字节）浮点数值的形式，它的范围在负数的时候是从–1.79769313486231E308～–4.94065645841247E-324，而在正数的时候是从 4.94065645841247E-324～1.79769313486232E308。Double 的类型声明字符是数字符号（#）。

若程序中处理的数据范围很大，或小数点后的位数较多，就应该采用双精度浮点型。

5. 货币型（Currency）

由名称可知该种数据类型主要用来保存货币值。货币型数据存储为 64 位（8 个字节）整型的数值形式，然后除以 10000 给出一个定点数，其小数点左边有 15 位数字，右边有 4 位数字。这种表示法的范围可以从–922337203685477.5808～922337203685477.5807。货币型的类型声明字符为 at 符号（@）。

货币型数据类型在货币计算与定点计算中很有用，在这种场合精度特别重要。浮点（单精度和双精度）数据比货币型的有效范围大得多，但有可能产生小的进位误差，而货币型采用更多的字节保存数据，能减少计算的误差。

6．字节型（Byte）

字节型数据类型为数值型，用来保存 0～255 之间的整数，占用 8 位（1 个字节）存储空间。字节型数据类型在存储二进制数据时很有用。

字符型表示数值范围很限（只能为 0～255），所以，一般情况下都不使用这种类型保存数据。

7．字符串（String）

字符串是一个字符序列，类似于 Excel 中的文本。在 VBA 中，字符串包括在双引号内（半角状态的双引号）。其中长度为 0（双引号中不包括任何字符）的字符串称为空字符串。以下为字符串的表示形式：

```
"Microsfot Visual Basic"
"欢迎使用 VBA"
""
```

其中最后一个为空字符串。

VBA 中的字符串又分两种：变长与定长的字符串。

- ❑ 变长字符串的长度是不确定的，最多可包含大约 20 亿（2^31）个字符。
- ❑ 定长字符串的长度是确定的，可包含 1 到大约 64K（2^16）个字符。

以下语句声明字符串：

```
Dim str1 As String
Dim str2 As String*10
```

其中变量 str1 为变长字符串，可保存多个字符，而变量 str2 表示定长字符串，该变量定义后将一直占用 10 个字符位置。

8．布尔型（Boolean）

布尔型数据很简单，只有两个值（True 或 False）。该类型适合存储简单的二元信息，例如，真/假、是/否等类似的信息。布尔型变量的值显示为 True 或 False，保存为 16 位（2 个字节）的数值形式。

当转换其他的数值类型为 Boolean 值时，0 会转成 False，而其他的值则变成 True。当转换 Boolean 值为其他的数据类型时，False 成为 0，而 True 成为–1。

9．日期型（Date）

在 VBA 中支持复杂的日期操作和运算。

日期型数据存储为 64 位（8 个字节）浮点数值形式，其可以表示的日期范围从 100 年 1 月 1 日到 9999 年 12 月 31 日，而时间可以从 0:00:00 到 23:59:59。任何可辨认的文本日期都可以赋值给 Date 变量。日期文字需以数字符号（#）括起来，例如，#January 1, 1993# 或 #1 Jan 93#。

日期型变量会根据计算机中的短日期格式来显示。时间则根据计算机的时间格式（12

小时制或 24 小时制）来显示。

当其他数值类型要转换为日期型时，小数点左边的值表示日期信息，而小数点右边的值则表示时间。午夜为 0 而中午为 0.5。负整数表示 1899 年 12 月 30 日之前的日期。

10. 对象型（Object）

VBA 是面向对象的程序设计语言，用户可在程序中访问各种对象，例如，Excel 的工作表、单元格等。这些对象有自己特定的对象名，这里所说的对象型可引用任何对象。

对象型数据存储为 32 位（4 个字节）的地址形式，其为对象的引用（即引用指定对象）。必须使用 Set 语句给对象变量赋值，对象变量使用结束后，应为其赋值为 Nothing。如：

```
Dim MyObject As Object
Set MyObject = Worksheets("sheet1")     ' 赋值对象引用工作表
Set MyObject = Nothing                  ' 中断关联
```

11. 变体型（Variant）

变体型是 VBA 中的一种特殊数据类型，所有没有被声明数据类型的变量都默认为变体型。变体型数据是所有没被显式声明（用如 Dim、Private、Public 或 Static 等语句——变量的声明将在本章后面进行介绍）为其他类型变量的数据类型。变体型没有类型声明字符。

变体型是一种特殊的数据类型，除了定长 String 数据及用户定义类型外，可以包含任何种类的数据。Variant 也可以包含 Empty、Error、Nothing 及 Null 等特殊值。可以用 VarType 函数或 TypeName 函数来决定如何处理变体型中的数据。

4.3.2 自定义数据类型

在 VBA 中，可以使用 Type 语句定义自己的数据类型。用户自定义类型经常用来表示数据记录，记录一般由多个不同数据类型的元素组成。

定义自定义数据类型的语法格式如下：

```
Type 数据类型名
    数据类型元素名 As 数据类型
    数据类型元素名 As 数据类型
      ……
End Type
```

其中：“数据类型名”就是要定义的数据类型的名字。“数据类型”为前面介绍的基本数据类型（也可使用用户已经定义的自定义类型）。例如：

```
Type Product
    ProductName As String           '产品名称
    Quantity As Integer             '库存数量
    Price As Currency               '单价
    Order As Integer                '订购量
End Type
```

有了以上的自定义类型，即可使用该类型的变量保存一行的数据。这样就可以非常方便地处理 Excel 工作表中的数据。

使用 Type 语句声明了一个用户自定义类型后，就可以在该声明范围内的任何位置声明该类型的变量。可以使用 Dim、Private、Public、ReDim 或 Static 来声明用户自定义类型的变量。

注意：自定义数据类型的定义必须放在模块（模块和类模块）的声明部分。在使用记录类型之前，必须用 Type 语句进行定义。一般情况下，记录类型在模块中定义，其变量可以出现在 VBA 工程的任何地方。

例如：如图 4-1 所示的工作表中每行数据产品名称、库存量、单价、订购量等数据。在 VBA 程序中为了方便地处理这些数据，可自定义数据类型 Product，然后在程序中使用 Dim 声明一个变量的数据类型为 Product，然后在程序中使用赋值语句将各列的值保存到声明的变量中，具体的代码如下：

销售管理.xls [兼容模式]

商品信息表

产品ID	产品名称	供应商ID	类别ID	单位数量	单价	库存量	订购量	再订购量
1	苹果汁	1	1	每箱24瓶	18	39	0	1
2	牛奶	1	1	每箱24瓶	19	17	40	2
3	蕃茄酱	1	2	每箱12瓶	10	13	70	2
4	盐	2	2	每箱12瓶	22	53	0	
5	麻油	2	2	每箱12瓶	21.35	0	0	
6	酱油	3	2	每箱12瓶	25	120	0	2
7	海鲜粉	3	7	每箱30盒	30	15	0	1
8	胡椒粉	3	2	每箱30盒	40	6	0	
9	鸡	4	6	每袋500克	97	29	0	

商品　员工　供应商　客户

图 4-1　商品信息表

```
Type Product
    ProductName As String     '产品名称
    Quantity As Integer       '库存量
    Price As Currency         '单价
    Order  As Integer         '订购量
End Type
Sub test()
    Dim p1 As Product
    With Worksheets("商品")
        p1.ProductName = .Cells(3, 2)
        p1.Price = .Cells(3, 6)
        p1.Quantity = .Cells(3, 7)
        p1.Order = .Cells(3, 8)
    End With
End Sub
```

4.3.3　枚举类型

枚举就是将变量的值逐一列举出来，属于该枚举型的变量只能取列举的某一个值。当一个变量只有几种可能的值时，可以定义为枚举类型。

在 Excel VBA 中预定义了很多枚举类型。例如在 Excel 中，对象的水平对齐方式共有 8 种，使用 xlHAlign 枚举类型来表示，其枚举名称、值和表示的意义如表 4-2 所示。

表 4-2 xlHAlign枚举类型

名　　称	值	描　　述
xlHAlignCenter	–4108	居中对齐
xlHAlignCenterAcrossSelection	7	跨列居中
xlHAlignDistributed	–4117	分散对齐
xlHAlignFill	5	填充
xlHAlignGeneral	1	按数据类型对齐
xlHAlignJustify	–4130	两端对齐
xlHAlignLeft	–4131	左对齐
xlHAlignRight	–4152	右对齐

枚举类型提供了一种处理有关常数的方法。例如，可以定义一个枚举类型来表示小学六个年级的名称与一组整型常数的关系，在代码中就可使用年级的名称（如“一年级”），而不用使用数值来表示了。这样程序将具有更好的阅读特性。枚举类型的定义需放在模块、窗体的声明部分，其定义格式如下：

```
Public | Private] Enum 类型名称
    成员 [= 常数表达式]
    成员 [= 常数表达式]
    ...
End Enum
```

说明：

在默认情况下，枚举中的第 1 个常数为 0，其后的常数比前面一个大 1，例如：



```
Public Enum Grade
    小学
    一年级
    二年级
    三年级
    四年级
    五年级
    六年级
End Enum
```

在以上定义中，第 1 项的值为 0，本例随意设置了一个值，使“一年级”可表示常数 1，“六年级”表示常数 6。

如果希望枚举型的第 1 项的值为常数 1，则可按以下方式进行定义：

```
Public Enum Grade
    一年级 = 1
    二年级
    三年级
    四年级
```

```
    五年级
    六年级
End Enum
```

这时，“一年级”表示常数 1，“二年级”表示常数 2。

在代码窗口的声明部分定义了枚举类型后，就可以声明该枚举类型的变量，并使用它。例如定义了上面的 Grade 枚举类型后，即可在代码窗口中使用该类型，在定义为变量时将显示前面定义的枚举类型 Grade，如图 4-2 所示。给枚举类型变量赋值时，在代码窗口中将自动列出枚举类型的成员，如图 4-3 所示。



图 4-2　定义枚举变量

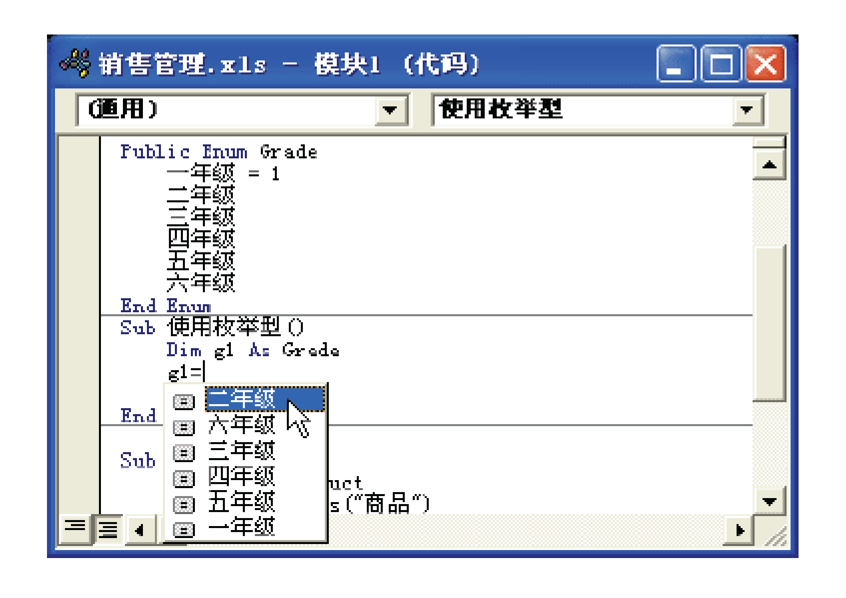


图 4-3　使用枚举型

4.4　常　　数

VBA 应用程序通过获取数据、处理数据、输出数据来完成用户的工作。在处理数据时，像 3.14、255 等仅表示它自身取值的数据叫常数。常数的值在程序执行之前就已经确定，执行过程中不能改变。

VBA 中包括直接常数、符号常数和系统常数 3 种。

4.4.1　直接常数

直接常数是指在 VBA 程序中可以直接使用的量。根据表示的数据类型不同，直接常数分为数值常数（整型、长整型、单精度型、双精度型、货币型等）、字符串常数、日期/时间常数和布尔常数等多种类型。

1. 数值常数

数值常数是由数字、小数点和正负符号所构成的量。一个数值常数有时可能存在多种数据类型的解释。例如，3.14 可解释为单精度型，也可为双精度型。VBA 将使用占用内存少的那种类型，即单精度型。

为了标识直接常数的类型，可在直接常数的后面加上类型标识符，例如：

```
s=r*r*3.14#
```

以上代码中，3.14#表示该常数值按双精度型存储，若 3.14 后面不带“#”号，则表示是单精度常数。

2．字符串常数

字符串常数是由数字、英文字母、特殊符号和汉字等可见字符构成的，在书写时必须使用半角状态的双引号作定界符，如：

```
"Excel VBA 从入门到精通"
```

注意：字符串常数必须用双引号括起来，否则，VBA 会将其认为是一个变量名。

3．日期/时间常数

在 Excel 中，可以方便地处理日期/时间数据。日期/时间常数用来表示某一天或某一具体时间。在输入日期/时间常数时必须使用“#”作为定界符，并且输入的日期时间要有具体的意义。例如，#8/8/2008#是正确的，而#4/31/2008#是错误的，因为 4 月没有 31 日。

输入日期时，VBE 自动将其两个“#”号中的数据转换为正确的日期格式。例如，在 VBE 的代码中输入以下内容：

```
#13/8/2008#
#2008/8/13#
```

VBE 都会自动将其按“月/日/年”的格式转换为正确的表示形式，显示如下：

```
#8/13/2008#
```

若输入的值如下：

```
#13/13/2008#
```

则将弹出如图 4-4 所示的错误提示信息。



图 4-4　错误提示

提示：日期值中的年份可省略世纪值，当省略世纪值时，其表示的年份与 Windows 操作系统中的设置相关。例如，#8/13/08#可能表示 2008 年 8 月 13 日。

4．布尔常数

布尔常数也叫逻辑常数。在 VBE 中布尔常数只有两个值：True（真）和 False（假）。

4.4.2　符号常数

符号常数是一种代替直接常数的标识符。如果在程序中需反复地使用某一个常数，可为该常数命名，在需要使用该常数的地方引用其常数名即可。

一般在 VBA 代码的上方定义符号常数，当程序中需要修改该常数的值时，只需要在

程序开始处定义一次符号常数即可。

在程序运行过程中，不能对符号常数进行赋值和修改，即符号常数在程序运行前必须有确定的值。VBA 中可使用 Const 关键字定义符号常数，其语法格式如下：

```
Const 常数标识符 As 数据类型 = 符号常数表达式
```

其中，Const 为定义符号常数的关键字，符号常数表达式计算出来的值保存在常数名中，如果在声明常数时没有显式地使用 As type 子句，则该常数的数据类型是最适合其表达式的数据类型。

例如，以下代码定义符号常数：

```
Const MAXCOL = 255
Const MyInt As Integer = 5
Const BOOKNAME = "Excel VBA 从入门到精通"
```

在定义符号常数时，等号右边的表达式往往是数字或字符串，也可以前面定义过的常数。如：

```
Const BOOK = "Microsoft Office " + BOOKNAME
```

以上代码使用前面定义的 BOOKNAME 符号常数，通过运算后得到新的符号常数 BOOK。

4.4.3　系统常数

为了控制 Excel 中的各对象，Excel VBA 预定义了许多常数，这些常数称为系统常数。提供对象库的其他应用程序（如 Access、Excel、Project 以及 Word 等）也提供常数列表，这些常数可与它们所属的对象、方法以及属性等一起使用。

在 VBA 中，系统常数名采用大小写混合的格式，其前缀表示定义常数的对象库名。在 Excel 中的系统常数名通常都是以小写的 xl（如 xlWindowType 的成员包括 xlWorkbook 等几个）作为前缀，而 VB 中的系统常数名通常都是以小写的 vb 作为前缀。

例如，VBA 中常用的 Msgbox 用来显示一个信息对话框，为了方便控制对话框，VBA 提供了很多 Msgbox 常数，用 vbOKOnly 表示对话框只有【确定】按钮，比用数值 0 更直观易用。

要查询某个系统常数的具体名称及其确切值，可通过【帮助】菜单或使用【对象浏览器】窗口来查看。

通过【帮助】菜单查看常数值的方法如下：

（1）在 VBE 中按 F1 打开帮助系统。

（2）在【搜索】文本框中输入关键字“常数”，单击【搜索】按钮，将显示关键字为“常数”的所有帮助条目，如图 4-5 所示。

（3）在查找到的信息中单击【Visual Basic 常数】按钮，可显示 VBA 常数的分类，如图 4-6 左图所示。单击分类【Color 常数】按钮即可查看具体的常数，如图 4-6 右图所示。

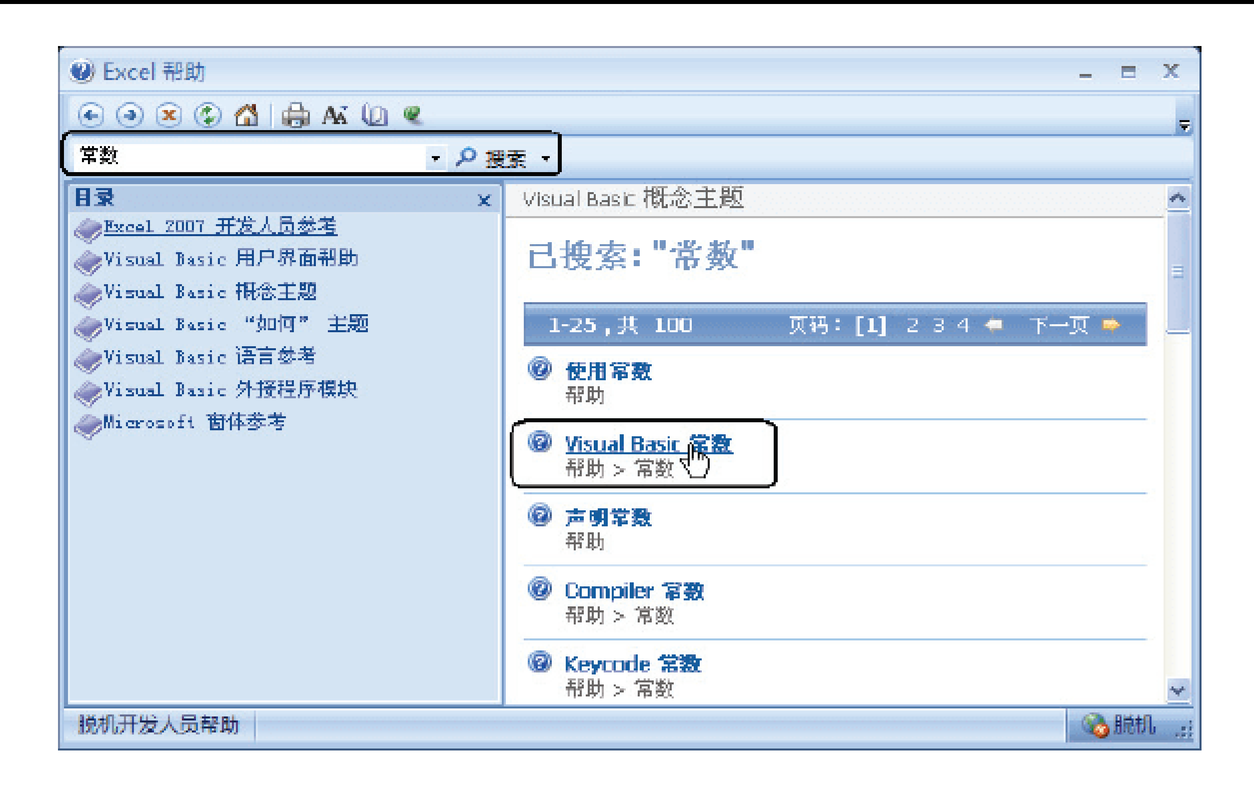


图 4-5　在【帮助】菜单中查看常数



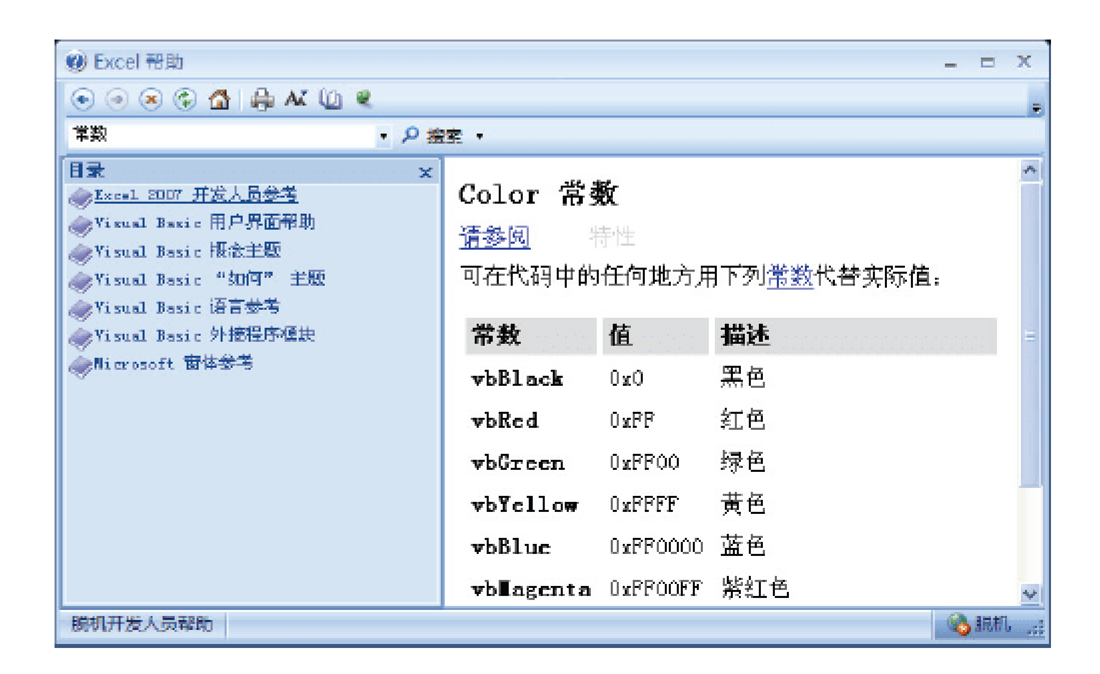


图 4-6　查看常数

还可通过【对象浏览器】窗口查找系统常数，具体步骤如下：

（1）在 VBE 中按 F2 打开【对象浏览器】窗口。

（2）在该窗口上部第一个列表框中选择相应的库，在第二行的列表框中输入要查找的常数，单击【搜索】按钮，将显示 Msgbox 的相关常数，如图 4-7 所示。

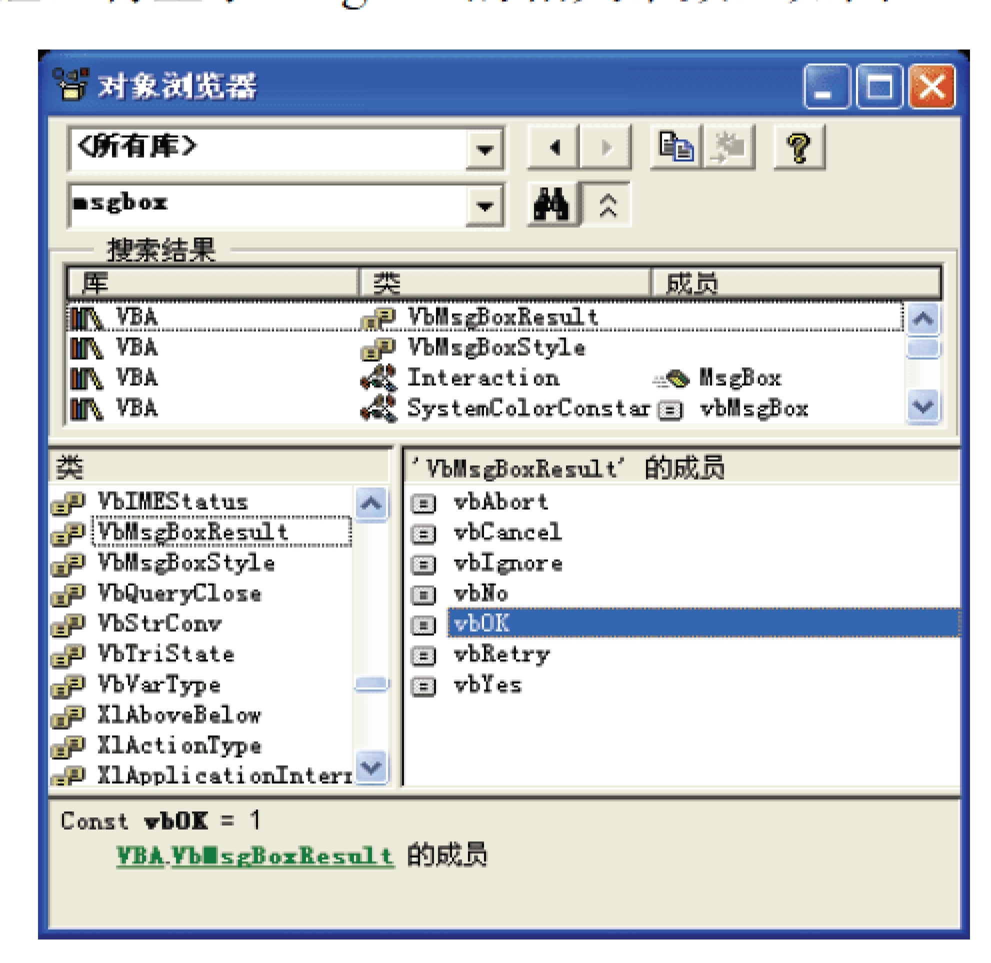


图 4-7　【对象浏览器】窗口

4.5　变　　量

变量用于保存程序运行过程中的临时值，根据其保存的数据，变量也具有不同的类型。和常数不同，在程序运行过程中，变量保存的值可以进行更改。

4.5.1　声明变量

声明变量就是事先将变量名及其类型在使用之前通知 VBA，由 VBA 按照变量的数据类型分配存储空间。可使用 Dim、Static、Private 或 Public 关键字来声明变量。最常用的是使用 Dim 声明变量，其语法格式如下：

```
Dim 变量名 [As 数据类型]
```

其中：

- Dim 和 As 为声明变量的关键字。
- 数据类型为前面介绍的类型关键字，例如，String、Date 等。
- 中括号部分表示可以省略，即声明变量时也可不指定变量的类型。

可以在一个语句中声明几个变量。而为了指定数据类型，必须将每一个变量的数据类型包含进来。在下面的语句中，变量 intX、intY 与 intZ 被声明为 Integer 类型。

```
Dim intX As Integer, intY As Integer, intZ As Integer
```

在 VBA 中变量的声明分隐式声明和显示声明两种。

1. 隐式声明

在使用一个变量之前不必先声明这个变量。这种变量使用方式称为隐式变量声明。使用隐式变量时，VBA 会自动创建变量，并设置为 Variant 类型。在为其指定值之前，其值为 Empty；当为它赋值后，会采用所赋值的类型作为变量的类型。

使用隐式声明的方法，看起来很方便。但是，当程序很大或很复杂时，这种未经声明的变量使用时往往会造成程序出错（如变量名拼写错误），而这种错误不能利用编译系统检查出来，大量的未声明变量的检查工作往往靠人工逐个检查，从而增加调试的难度。因此，建议在使用每个变量之前要先声明，这就是变量的显式声明方式。

2. 显式声明

为了避免隐式声明引起的麻烦，可以规定，只要遇到一个未声明的变量名，VBA 就发出错误警告。要显式声明变量，可以在模块、类模块的声明段中加入以下语句：

```
Option Explicit
```

4.5.2 变量的作用域和生存期

在 VBA 应用程序中，将使用很多的变量，有些变量可在整个应用程序中使用，而另一些变量只能在某一个过程中使用，这就是变量的作用域。

有的变量在过程使用完毕后即自动消失，而有的变量再次进入过程时还可访问其值，这就是变量的生存期。

1. 变量的作用域

变量的作用域是指变量可以被程序使用的范围，变量的作用域共分为以下 3 种。

- 过程级别：称为局部变量。只能在过程中使用的变量，过程执行结束后，变量的值自动消失。这类变量用 Dim 关键字进行定义。
- 模块级别：称为模块变量。在 Excel VBA 的某个模块顶端定义的变量，在该模块的各过程中都可访问模块级别变量。这类变量在模块的开始部分使用 Dim 或 Private 关键字进行定义。
- 全局级别：称为全局变量。在整个应用程序范围内（各模块、各过程）都可以使用的变量。这类变量在模块的开始部分用 Public 关键字进行定义。

2. 变量的生存期

变量不仅有作用域，还有生存期。一般情况下，变量的生存期与其作用域相同，即变量在其作用域内有效，超出作用域后变量的值自动消失。

在过程中使用 Static 关键字声明的变量称为静态变量，这类变量在整个应用程序中有效，即应用程序未结束，则变量的值一直保存在内存中，且其内容不会改变。但静态变量只在定义的过程中才能访问，超过该过程时，静态变量将不能被访问。

4.5.3 局部变量

局部变量也只有局部作用域，它在程序运行期间不是一直存在，而是只在定义的过程中存在，执行过程结束后，变量被撤销，其所占用的内存也被收回。

将定义变量的 Dim 语句放到过程中，可创建属于过程级别的局部变量。例如，以下代码创建了变量 strName 并且指定为 String 数据类型。

```
Dim strName As String
```

在过程中定义的变量 strName 只可以在此过程中被使用。

下面以实例演示局部变量的作用域。在模块中输入以下代码：

```
Sub 过程1()
    Dim s1 As String
    s1 = "测试局部变量"
    MsgBox s1
End Sub
```

```
Sub 过程2()
    MsgBox s1
End Sub
```

上面的代码定义了两个过程“过程 1”和“过程 2”，其中“过程 1”中定义了一个局部变量 s1，并调用 MsgBox 显示该变量的值，如图 4-8 左图所示。在“过程 2”中直接使用 MsgBox 显示 s1 中的值，因为该过程中并未定义变量 s1 的值，所以显示对话框中无任何信息，如图 4-8 右图所示。





图 4-8 测试局部变量

4.5.4 模块变量

可以使用 Private 语句声明私有的模块级别变量。模块变量只可使用于同一模块中的过程。在模块级别中使用 Dim 语句与使用 Private 语句是相同的。不过使用 Private 语句可以更容易地读取和解释代码。使用两个语句声明变量的语法格式完全相同。

要声明模块变量，可按以下步骤进行：

（1）在 VBE 中双击模块名，打开模块的代码窗口。

（2）在模块的【声明】部分，输入声明变量的代码，如图 4-9 所示。

```
Private strName As String
```

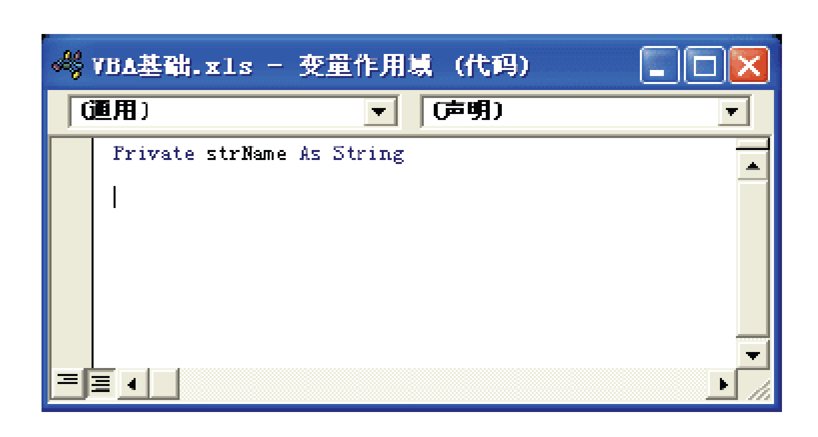


图 4-9 声明模块变量

（3）接着在该模块的下方输入以下两个过程，用来测试模块变量。

```
Sub 测试模块变量()
    strName = "伍远高"
    MsgBox "软件设计：" & strName
End Sub
Sub 显示模块变量的值()
    MsgBox strName
End Sub
```

注意：模块变量的初始化操作必须在过程中进行，不能在模块【声明】部分进行。

运行过程“测试模块变量”，将首先初始化模块变量 strName 的值，然后用 MsgBox 显示模块变量 strName 的值。接着运行过程“显示模块变量的值”，在该过程中未对模块变量 strName 赋值，但因 strName 为模块变量，在“测试模块变量”过程中已经赋值，所以也可显示出相同的值来。

如果首先运行过程“显示模块变量的值”，MsgBox 对话框将不会显示任何值。

4.5.5 全局变量

使用 Public 语句声明公共模块级别变量。全局变量可用于工程中的任何过程。如果全局变量是声明于标准模块或是类模块中，则它也可以被任何引用到此全局变量所属工程的工程中使用。

全局变量在整个应用程序范围内都有效，一般用来定义一些全局参数，如应用程序名称、程序版本等。

注意：一般情况下，要尽量少使用全局变量。过多地使用全局变量，可能会导致程序混乱。

下面演示全局变量的使用。

（1）在 VBE 中双击模块“变量作用域”，在模块的声明部分使用以下代码声明全局变量。

```
Public strAppName As String
```

（2）在该模块中编写如下过程，初始化全局变量，如图 4-10 所示。

```
Sub 初始化全局变量()
    strAppName = "Excel 测试应用程序"
    MsgBox strAppName
End Sub
```

（3）单击主菜单【插入】|【模块】命令，向工程中新增加一个模块，为模块设置名称为“全局变量”。

（4）在“全局变量”模块中编写以下过程，显示全局变量的值，如图 4-11 所示。

```
Sub 显示全局变量的值()
    MsgBox strAppName
End Sub
```

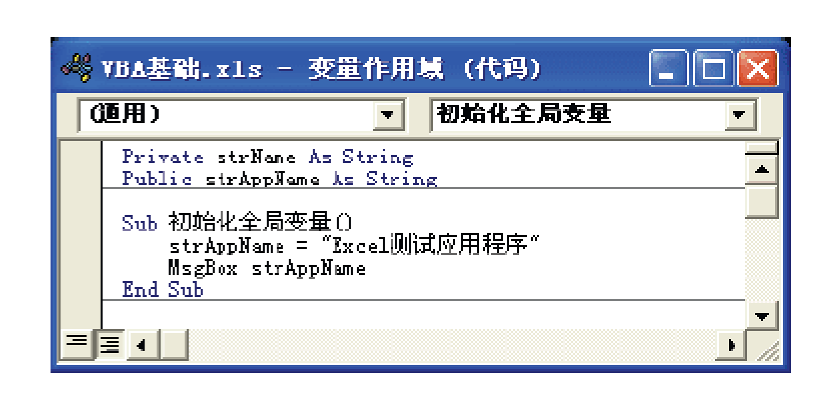


图 4-10 声明全局变量

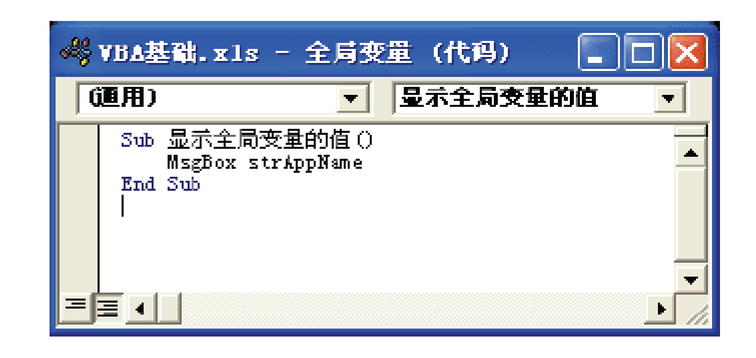


图 4-11 在其他模块中访问全局变量

（5）分别执行以上两个模块中的过程，可查看全局变量在不同模块中的值。

4.5.6　静态变量

将过程级变量（局部变量）声明为静态变量，可在过程退出时仍将该变量的值保留在内存中。在下次调用该过程时，又可访问上次退出时保留在内存中的值。

静态变量的声明方法是，在过程内部用 Static 关键字声明一个或多个变量，其用法和 Dim 语句完全一样，例如：

```
Static intCount As Integer
```

下面用实例来演示静态变量和局部变量生存期的状态。

（1）在 VBE 环境，单击主菜单【插入】|【模块】命令，向工程中插入一个模块。

（2）将模块名称改为“静态变量”。

（3）在模块中输入以下程序代码：

```
Sub 静态变量测试()
    Dim i1 As Integer
    Static i2 As Integer
    i1 = i1 + 1
    i2 = i2 + 1
    MsgBox "局部变量的值：" & i1 & vbNewLine & _
       "静态变量的值：" & i2, vbOKOnly
End Sub

Sub 测试()
    For i = 1 To 5
       Call 静态变量测试
    Next
End Sub
```

（4）执行过程“测试”，首先将显示如图 4-12 左图所示的结果，由图可看出局部变量和静态变量的值都为 1。单击【确定】按钮将再次显示类似的对话框，单击多次【确定】按钮后，可看到局部变量的值仍然为 1，而静态变量的值已变为了 5，如图 4-12 右图所示。





图 4-12　局部变量和静态变量

在以上代码中编写了两个过程，过程“测试”中循环调用 5 次“静态变量测试”过程，在“静态变量测试”过程中将变量 i1 声明为局部变量，该过程执行时进行初始化，执行完后就自动释放，所以，无论该过程执行多少次，输出的结果都为 1。

而变量 i2 声明为静态变量，只有第一次进入过程时才进行初始化操作，当退出该过程时并不释放 i2 变量，再次进入该过程后，i2 变量为原来保存的值，所以每执行一次过程，

变量 i2 的值就增加 1。

4.6 运算符和表达式

一个表达式由操作数和运算符共同组成。表达式中作为运算的数据称为操作数，操作数可以是常数、变量、函数或表达式；运算符是介于操作数间的运算符号，如 + 、– 都是典型的运算符。在 VBA 中提供了 4 种基本的运算，即算术运算、比较运算、逻辑运算和连接运算。由这些运算符连接操作数可组成 4 种不同的表达式。

4.6.1 算术表达式

由算术运算符连接组成的表达式称为算术表达式。表 4-3 列出了 VBA 中的算术运算符及其优先级。

表 4-3 算术运算符及其优先级

算术运算符	优 先 级	作 用	例 子
^	1	指数运算	2^2=4，3^2=9
–	2	负数运算	–6，–10
*、/	3	乘、除	2*4=8，7/2=3.5
\	4	整除	7\2=3，8\3=2
MOD	5	取模	7\2=1，8\3=2
+、–	6	加、减	2+4=6，8-2=6

在表达式中，使用运算符对操作数进行计算时都有一定的顺序（如，在算术运算中先要算乘除后算加减）。优先级和结合性是运算符两个重要的特性，结合性又称为计算顺序，它决定组成表达式的各个操作数是否参与计算以及什么时候计算。例如：

```
3+2*5
```

以上表达式中，将先计算 2*5，将得到的结果再与 3 相加，得到最终结果为 13。

在表达式中，当有两个优先级相同的运算符，则从左向右进行运算。如果要改变表达式的运算优先级，可以使用括号。例如：

```
（3+2）*5
```

以上表达式将先计算 3+2 的和，再与 5 相乘，得到最终结果为 25。

4.6.2 比较表达式

比较运算又称为关系运算，用来比较两个操作数，其运算结果为逻辑值，只能为 True 或 False 两种可能。表 4-4 列出了比较运算符的功能，比较运算符的优先级相同。

表 4-4　比较运算符的功能

比较运算符	作　　用	例　　子
=	等于	x=y
<>	不等于	x<>y
<	小于	x<y
>	大于	x>y
<=	小于或等于	x<=y
>=	大于或等于	x>=y

例如，运行以下代码，将在【立即窗口】显示如图 4-13 所示的结果。

```
Debug.Print "A" > "B"
Debug.Print "A" > "a"
Debug.Print 2 <= 5
```

在使用比较运算符时，对于数值型的数据，其大小比较很好理解。而对于字符串的比较，在 VBA 中有特殊的规定。

比较字符串时，按字母的 ASCII 码进行比较。例如，字母 A 的 ASCII 码为 65，而字母“B”的 ASCII 码为 66，因此以下语句：

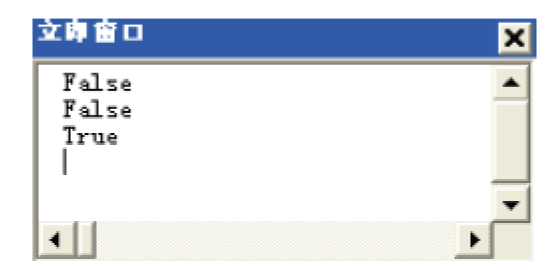


图 4-13　立即窗口

```
Debug.Print "A" > "B"
```

输出的结果为 False，即字母 A 小于字母 B。

当比较不同类型的数据时，有可能产生错误，例如：

```
Debug.Print 2 > "A"
```

当 VBA 执行以上语句时，因数值 2 和字母 A 为不同类型，所以二者进行比较时将弹出如图 4-14 所示的错误提示信息。

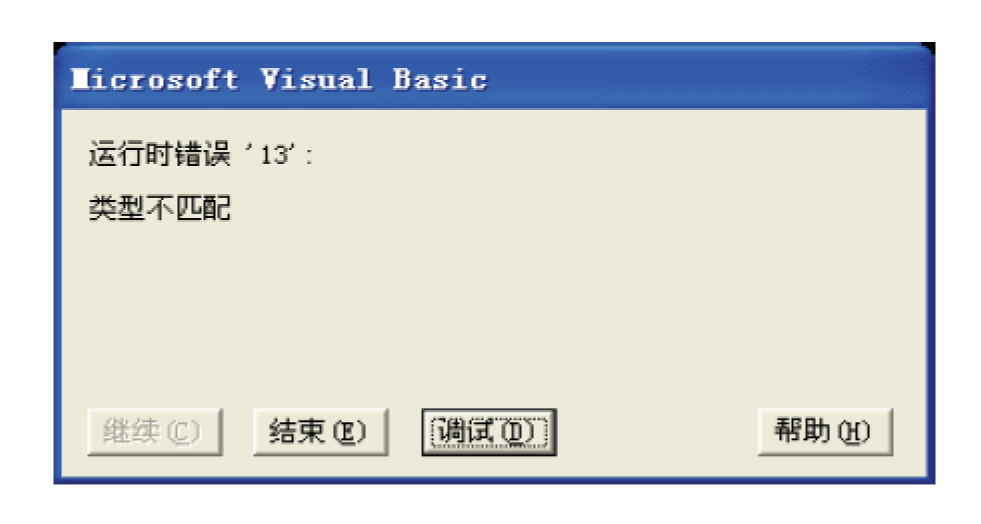


图 4-14　错误提示

4.6.3　逻辑表达式

逻辑运算是指表达式间的逻辑关系，其运算结果为逻辑值，只能为 True 和 False 两种可能。表 4-5 列出了逻辑运算操作符及其优先级。

表 4-5　逻辑运算操作符及其优先级

逻辑运算符	优　先　级	作　　用	例　　子
Not	1	取反运算	Not X
And	2	逻辑与运算	X And Y
Or	3	逻辑或运算	X Or Y
Xor	4	逻辑异或运算	X Xor Y
Eqv	5	逻辑相等运算	X Eqv Y
Imp	6	逻辑蕴涵运算	X Imp Y

如果以 A 和 B 代表任意两个操作数，而以 T 代表逻辑真 True，以 F 代表逻辑假 False，则各种逻辑运算符的运算结果如表 4-6 的真值表所示。

表 4-6　逻辑运算真值表

操作数 A	F	F	T	T
操作数 B	F	T	F	T
Not A	T	T	F	F
A And B	F	F	F	T
A Or B	F	T	T	T
A Xor B	F	T	T	F
A Eqv B	T	F	F	T
A Imp B	T	T	F	T

4.6.4　连接运算表达式

字符串连接运算的作用是用来连接两个以上的字符串，使其成为一个单一字符串。例如：str1="Excel "+"VBA"（字符串 Excel 后面有一个空格），其结果相当于 str1="Excel VBA"。

连接运算符只有两个，即“+”和“&”。其区别为：

- ❑ “+”运算符连接两个操作数都是字符串的情况。
- ❑ “&”运算符是将两个操作数强制为字符串连接起来。

连接运算符在 MsgBox 中使用得很多，例如：

```
MsgBox "局部变量的值：" & i1 & vbNewLine & "静态变量的值：" & i2, vbOKOnly
```

以上代码将字符串和变量 i1、i2 的值连接起来，再显示到对话框中。

第 5 章　程序控制结构

与其他程序设计语言相同，VBA 中的程序代码也可按一定的顺序执行。在程序中有时需要选择某一部分代码执行，有时需要重复执行某一段代码。VBA 中提供了完善的程序结构控制代码来完成这些功能。

5.1　VBA 程序结构概述

在 VBA 的程序代码中，语句是构成程序的基本成分，是程序的主体部分。每个语句以 Enter 键结束。

程序控制结构代码也是一些特殊的语句，这些语句用来控制程序语句的执行顺序。

5.1.1　认识语句

默认情况下，在 VBE 中输入语句后，VBE 将自动进行语法检查，如果发现语法错误，将打开一个提示对话框。

1．自动格式化

输入 VBA 语句后，VBE 将按一定的规则进行简单的格式化处理。例如，将关键字的首字母大写，在运算符前后加入空格，删除各部分多余的空格等。

开发人员在输入 VBA 关键字时，可以不区分大小写。例如输入 MsgBox 时，无论输入的是 Msgbox、msgbox，还是 MSGBOX，当输入完该函数的参数并按 Enter 键后，VBE 自动将其变为 MsgBox。

为了提高程序的可读性，在 VBA 代码中应加上适当的空格。当按 Enter 键完成语句的输入后，各关键字之间无论插入多少空格，VBE 都将其自动调整为一个空格。例如，输入以下代码（在关键字“worksheets("sheet1")”与“range("a1")”之间插入了多个空格，在“=”与“0”之间无空格）：

```
worksheets("sheet1").  range("a1")=0
```

输入完语句并按 Enter 键后，VBE 将自动格式化以上语句，得到如下代码：

```
Worksheets("sheet1").Range("a1") = 0
```

在“=”前后各插入一个空格，其他关键字之间的空格被自动删除。

🔔注意：不能在关键字的中间加入空格。

2. 复合语句与语句断行

一般情况下，VBA 程序中每个语句占一行。但在 VBA 中，也可以把几个语句放在一行中构成复合语句。各语句之间用冒号（：）分隔，例如：

```
Dim strName As String: strName = "伍远高"
```

与以下两行语句功能相同：

```
Dim strName As String
strName = "伍远高"
```

在 VBE 的代码窗口中，每行 VBA 代码可包含 1023 个字符。但是，为了使程序便于阅读，建议读者将一条长的语句打断为两行或多行。

VBA 中使用空格后接着一个下划线——续行符，可将一行代码延伸成两行以上。

例如，以下语句：

```
MsgBox "局部变量的值：" & i1 & vbNewLine & "静态变量的值：" & i2, vbOKOnly
```

可改写为以下格式：

```
MsgBox "局部变量的值：" & i1 & vbNewLine _
    & "静态变量的值：" & i2, vbOKOnly
```

通过用续行符（_）可创建长的逻辑行。一个逻辑行，最多可包含 24 个连续的续行字符，也就是最多可以包含 25 个物理行。这样，逻辑行的字符总量可达 10230 字符。如果超过了，必须将该行分为若干语句，或指定一些表达式为中间变量。

5.1.2 结构化程序设计的控制结构

结构化程序的概念首先是从以往编程过程中无限制地使用转移语句而提出的。使用 VBA 的 GoTo 转移语句可以使程序的控制流程强制性地转向程序的任一处。如果一个程序中多处出现这种转移情况，将会导致程序流程无序可寻，程序结构杂乱无章，这样的程序是令人难以理解和接受的，并且容易出错。尤其是在实际软件产品的开发中，更多的是追求软件的可读性和可修改性，像这种结构和风格的程序是不允许出现的。

为此提出了程序的三种基本结构，即程序的顺序、选择和循环三种控制流程，这就是结构化程序设计方法强调使用的三种基本结构。任何简单或复杂的算法都可以由这三种基本结构组合而成。所以，这三种结构就被称为程序设计的三种基本结构。也是结构化程序设计必须采用的结构。

- ❑ 顺序结构：就是按照语句的书写顺序从上到下、逐条语句地执行。执行时，排在前面的代码先执行，排在后面的代码后执行，执行过程中没有任何分支。这是最普遍的结构形式，也是后面两种结构的基础。
- ❑ 选择结构：又叫分支结构。是根据“条件”来决定选择执行哪一分支中的语句。

包括二分支和多分支，分支的嵌套。

- 循环结构：循环结构的程序设计比分支结构复杂。循环结构的思想是利用计算机高速处理运算的特性，重复执行某一部分代码，以完成大量有规则的重复性运算。

结构化程序中的任意基本结构都具有唯一入口和唯一出口，并且程序不会出现死循环。在程序的静态形式与动态执行流程之间具有良好的对应关系。

5.2 常 用 语 句

顺序结构就是从头到尾依次按顺序逐条执行语句，不需要控制语句。本节介绍 VBA 程序设计中常用的语句，如赋值、输入、输出语句等的用法。

5.2.1 赋值语句

使用赋值语句可以更改变量的值。赋值，顾名思义，就是把一个值赋予某个量。可以这样理解：变量相当于装东西的容器，赋值的过程就是把东西放进容器的过程。赋值语句格式如下：

```
[Let] 变量名 = 表达式
```

赋值语句首先对表达式进行运算，并将运算结果赋给左侧的变量或属性。

在 VBA 中可省略赋值关键字 Let，变量名必须遵循标识符的命名约定。

只有当表达式是一种与变量兼容的数据类型时，该表达式的值才可以赋给变量或属性。不能将字符串表达式的值赋给数值变量，也不能将数值表达式的值赋给字符串变量。如果这样做，就会在编译时出现错误。

可以用字符串或数值表达式赋值给 Variant 变量，但反过来不一定正确。任何除 Null 之外的 Variant 都可以赋给字符串变量，但只有当 Variant 的值可以解释为某个数时才能赋给数值变量。可以使用 IsNumeric 函数来确认 Variant 是否可以转换为一个数。

使用赋值语句时需注意以下两点：

- 赋值号（=）和比较运算符中的（=）的含义是不同的。赋值号是将表达的计算结果保存到左边的变量中。所以，赋值号左边必须为一个变量，而不能是表达式，例如：

```
i+j=10
```

VBA 将以上语句解释为比较表达式，而不是赋值语句。

- 在进行赋值操作时，正常情况下表达式结果的类型与变量数据类型是相同的，不需要进行任何类型转换。如果类型不一致，一般将表达式结果转换为变量的类型。

有关赋值时进行类型转换的演示代码如下：

```
Sub 赋值数据类型转换()
    Dim i As Integer
```

```
    Dim s As String
    Dim b As Boolean
    Dim v

    '将单精度数据赋给整型变量
    i = 3.14
    Debug.Print "整型变量 i="; i
    '将单精度数据赋给字符串变量
    s = 3.14
    Debug.Print "字符串变量 s="; s

    '将单精度数据赋给逻辑变量
    b = 3.14
    Debug.Print "逻辑变量 b="; b
    '将单精度数据赋给 Variant 变量
    v = 3.14
    Debug.Print "可变变量 v="; v
End Sub
```

执行以上过程，在【立即窗口】列表框中显示的结果如图 5-1 所示。

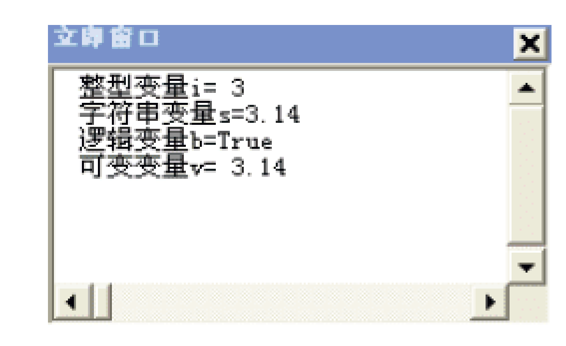


图 5-1　赋值类型转换

由图 5-1 的执行结果可知道，将单精度类型的数据赋值给整型变量时，将只取值的整数部分；将单精度类型数据赋值给字符串变量时，将表达式中的数据值转换为字符串型；将数值型数据赋值给逻辑变量时，非 0 值将转换为 True，0 将转换为 False。

5.2.2　注释语句

在程序中使用注释语句，可用文字描述程序的算法、函数中参数的意义、函数返回值的意义等，方便多个开发人员阅读代码，提高程序的可读性，方便代码的维护。

在 VBA 中，注释以撇号（'）开头，或者以 Rem 关键字开头，再在其后写上注释内容。Rem 语句的格式如下：

```
Rem 注释文本
```

也可以使用如下语法：

```
' 注释文本
```

在 Rem 关键字与“注释文本”之间必须要有一个以上的空格。

使用注释语句时需要注意以下几点：

- 注释语句是非执行语句，仅对程序的有关内容起注释作用。
- 注释语句可以使用任何字符。
- 注释语句不能放在续行符后面。
- 若使用撇号来添加注释文本，则在其他语句行后面使用时不必加冒号。

在 VBE 中，【编辑】工具栏提供了两个按钮：【设置注释块】和【解除注释块】，

使用这两个命令按钮可将选中的代码快速设置为注释，或取消其前面的注释符号（撇号）。如图 5-2 所示。

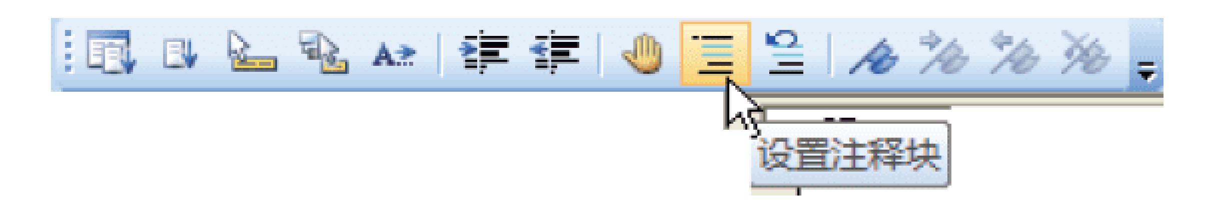


图 5-2　设置注释块

技巧：在调试程序时，可在不希望执行的代码前面添加注释符号。

5.2.3　使用 InputBox 输入对话框

在 VBA 中，开发人员可调用以下两种类型的 InputBox 输入对话框：

- 一种是 VB 中常用的 InputBox 函数，在该输入对话框中显示提示信息，等待用户输入正文或按下【确定】或【取消】按钮，并返回包含文本框的内容，该对话框返回值类型为字符串，如图 5-3 所示。
- 一种是使用 Application 对象的 InputBox 方法显示一个输入对话框，在该对话框中设置输入值的类型，如图 5-4 所示。



图 5-3　InputBox 函数对话框



图 5-4　InputBox 方法对话框

1．显示InputBox函数对话框

使用 VBA 提供的 InputBox 函数，可产生一个输入对话框。该函数将打开一个对话框作为输入数据的界面，等待用户输入数据，并返回所输入的内容。其语法格式如下：

```
InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])
```

该函数有 7 个参数，其意义分别如下所述。

- Prompt：为对话框消息出现的字符串表达式。其最大长度为 1024 个字符。如果需要在对话框中显示多行数据，则可在各行之间用回车换行符来分隔，一般使用 VBA 的常数 vbCrLf 代表回车换行符。
- Title：为对话框标题栏中的字符串。如果省略该参数，则把应用程序名放入标题栏中。
- Default：为显示在文本框中的字符串。如果省略该参数，则文本框为空。
- Xpos，Ypos：这两个参数必须成对出现，指定对话框的左上角坐标位置。如果省略该参数，则对话框会在水平方向居中。

- ❑ Helpfile：设置对话框的帮助文件，该参数可省略。
- ❑ Context：设置对话框的帮助主题编号，该参数可省略。

例如，使用以下代码可接收用户输入的数据。

```
Sub 使用 InputBox 函数()
    Dim sPrompt As String
    Dim sTitle As String
    Dim sDefault As String
    Dim sReturn As String
    sPrompt = "请输入用户姓名："
    sTitle = "输入姓名"
    sDefault = "伍远高"
    sReturn = InputBox(sPrompt, sTitle, sDefault)
    Debug.Print sReturn
End Sub
```

执行上面的代码，将显示如图 5-5 所示的对话框。

在文本框中输入用户姓名，单击“确定”按钮，程序将把用户输入的内容输出到【立即窗口】的列表框中。

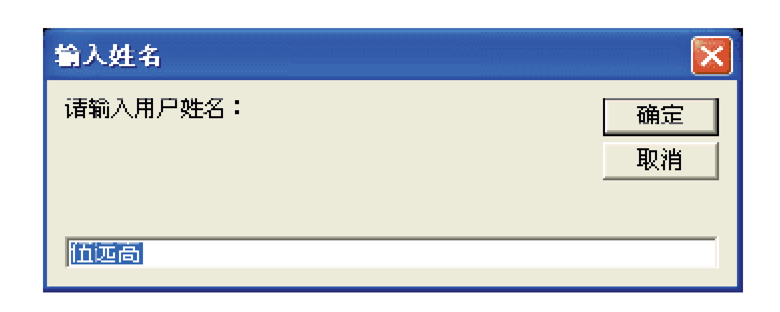


图 5-5　InputBox 函数

在使用 InputBox 函数时，应注意以下几点：

- ❑ 在默认情况下，InputBox 函数的返回值是一个字符串类型，而不是变体类型。如果需要使用该函数输入数值，则需要使用 Val 函数（或其他的转换函数）将返回值转换为相应类型的数值。
- ❑ 在对话框中，如果用户单击【取消】按钮（或按 Esc 键），则表示不使用当前输入的值，函数将返回一个空字符串。根据这一特性，可以判断用户是否输入了数据到对话框中。
- ❑ 执行一次 InputBox 函数，只能返回一个值，如果需要输入多个值，则必须多次调用该函数。

2. 显示InputBox方法对话框

使用 Application 对象的 InputBox 方法，也可显示一个接收用户输入的对话框。此对话框有一个【确定】按钮和一个【取消】按钮。如果单击【确定】按钮，则 InputBox 方法将返回对话框中输入的值。如果单击【取消】按钮，则 InputBox 方法返回逻辑值 False。

与 InputBox 函数不同的是，该对话框可指定输入数据的类型，具体的语法格式如下：

```
Application.InputBox(Prompt, Title, Default, Left, Top, HelpFile,
HelpContextID, Type)
```

从以上语法格式中可以看出，大部分参数与 InputBox 函数相同，只是在最后多了一个 Type 参数，用来指定输入数据的类型。Type 参数可设置为以下值之一或其中几个值的和。例如，对于一个可接受文本和数字的输入框，将 Type 设置为 1+2。

- ❑ 0：公式；

- 1：数字；
- 2：文本（字符串）；
- 4：逻辑值（True 或 False）；
- 8：单元格引用，作为一个 Range 对象；
- 16：错误值，如 #N/A；
- 64：数值数组。

例如，下面的代码提示用户在工作表 Sheet1 中选取一个单元格。

```
Sub 使用 InputBox 方法()
    Worksheets("Sheet1").Activate
    Set myCell = Application.InputBox( _
        prompt:="选择一个单元格", Type:=8)
End Sub
```

运行以上代码，将显示如图 5-6 所示对话框，用鼠标单击一个单元格，该单元格的引用地址将自动填入对话框的输入区域中。

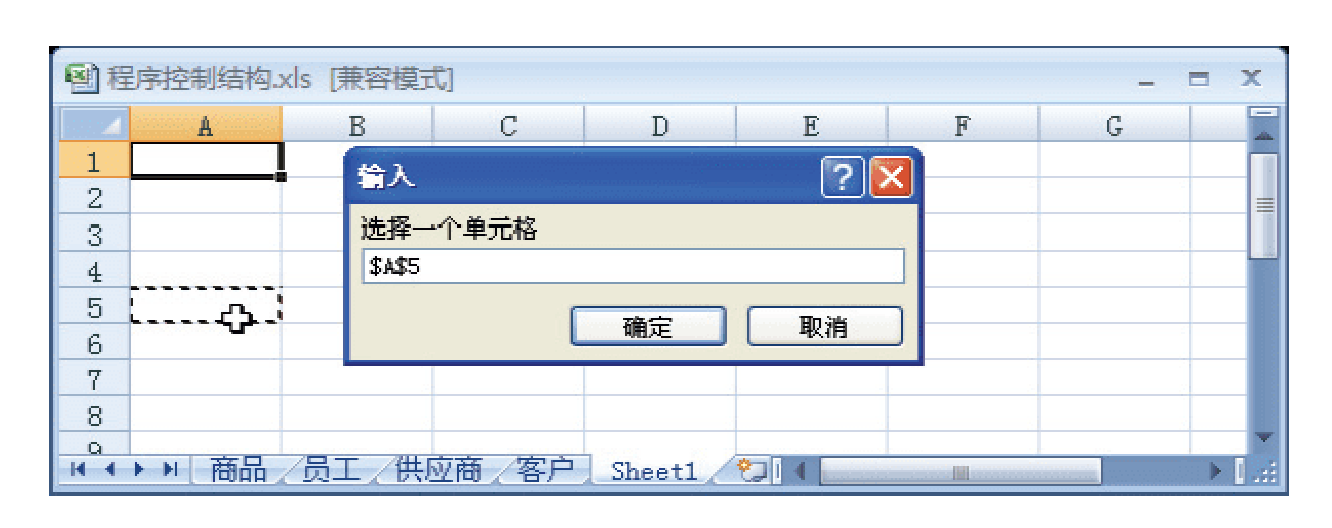


图 5-6　使用 InputBox 方法显示对话框

5.2.4　使用 MsgBox 函数显示信息

使用 MsgBox 函数可打开一个对话框，在对话框中显示消息，等待用户单击按钮，并返回一个整型值，告诉用户单击哪一个按钮。

MsgBox 函数的语法格式如下：

```
Value=MsgBox(prompt[,buttons][,title][ ,helpfile,context])
```

该函数有 5 个参数，除第 1 个参数外，其余参数都可省略。各参数的意义与 Inputbox 函数各参数的意义相同。不同的是多了一个 buttons 参数，用来指定在对话框中显示按钮的数目及形式、使用提示图标样式、默认按钮以及消息框的强制响应等。其常数值如表 5-1 所示。

表 5-1　按钮常数值

常　量	值	说　明
vbOkOnly	0	只显示“确定”（OK）按钮
vbOkCancel	1	显示“确定”（OK）及“取消”（Cancel）按钮

续表

常　　量	值	说　　明
vbAbortRetryIgnore	2	显示“异常终止”（Abort）、“重试”（Retry）及“忽略”（Ignore）按钮
vbYesNoCancel	3	显示“是”（Yes）、“否”（No）及“取消”（Cancel）按钮
vbYesNo	4	显示“是”（Yes）及“否”（No）按钮
vbRetryCancel	5	显示“重试”（Retry）及“取消”（Cancel）按钮
vbCritical	16	显示 Critical Message 图标
vbQuestion	32	显示 Warning Query 图标
vbExclamation	48	显示 Warning Message 图标
vbInformation	64	显示 Information Message 图标
vbDefaultButton1	0	以第 1 个按钮为默认按钮
vbDefaultButton2	256	以第 2 个按钮为默认按钮
vbDefaultButton3	512	以第 3 个按钮为默认按钮
vbDefaultButton4	768	以第 4 个按钮为默认按钮
vbApplicationModal	0	进入该消息框，当前应用程序暂停
vbSystemModal	4096	进入该消息框，所有应用程序暂停

表 5-1 中的数值（或常数）分为 4 类，其作用分别如下所述。

- 第一组值（0～5）：用来决定对话框中按钮的类型与数量。按钮共有 7 种，即确认、取消、终止、重试、忽略、是、否。每个数值表示一种组合方式。
- 第二组值（16，32，48，64）：用来决定对话框中显示的图标。共有 4 种，即暂停、疑问、警告、忽略。
- 第三组值（0，256，512，768）：设置对话框的默认活动按钮。活动按钮中文字的周转有虚线，按 Enter 键可执行该按钮的单击事件代码。
- 第四组值（0，4096）：决定消息框的强制响应性。

buttons 参数可由上面 4 组数值组成，其组成原则是：从每一类中选择一个值，把这几个值累加在一起就是 buttons 参数的值（大部分时间里都只使用前三组数值的组合），不同的组合可得到不同的结果。

例如：设置 buttons 参数为 16，得到的对话框如图 5-7 所示。设置 buttons 参数为 50，得到的对话框如图 5-8 所示。




图 5-7　MsgBox 效果一



图 5-8　MsgBox 效果二

MsgBox 函数除了显示提示信息之外，还可返回一个整数值，这个整数与所选择的按钮有关。MsgBox 函数显示的对话框有 7 种按钮，返回值与这 7 种按钮相对应，分别为

1～7 之间的整数，如表 5-2 所示。

表 5-2　MsgBox函数的返回值

常　数	值	描　述
vbOK	1	单击【确定】按钮
vbCancel	2	单击【取消】按钮
vbAbort	3	单击【终止】按钮
vbRetry	4	单击【重试】按钮
vbIgnore	5	单击【忽略】按钮
vbYes	6	单击【是】按钮
vbNo	7	单击【否】按钮

在 VBE 环境中输入 MsgBox 函数时，将自动列出常数值，选择了一个 buttons 常数后，输入加号将再次显示 buttons 的常数列表，如图 5-9 所示。

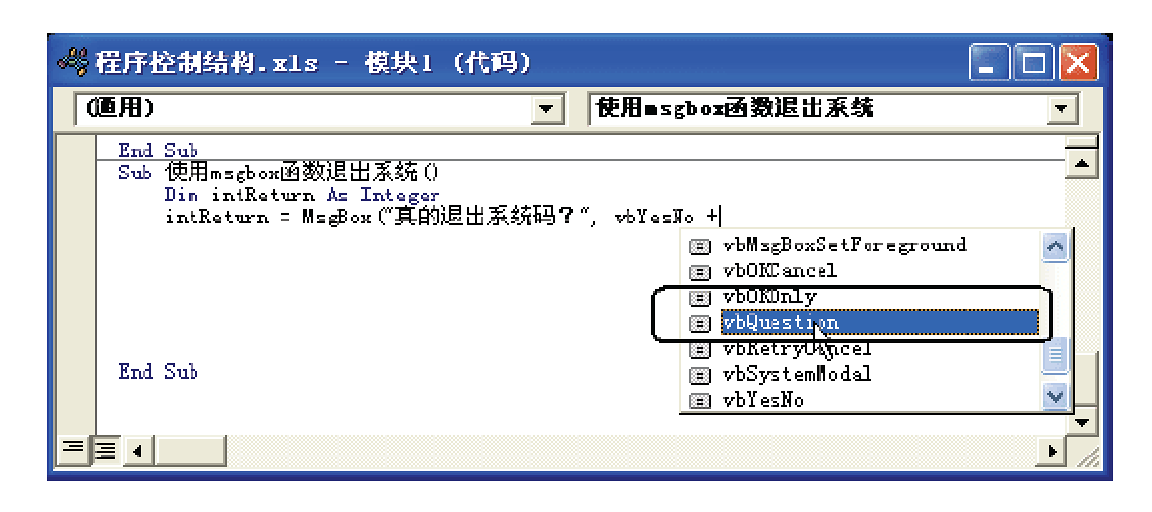


图 5-9　MsgBox 的图标常数列表

例如，使用下面的代码获取用户的选择，再根据用户的选择决定是否退出系统。

```
Sub 使用 msgbox 函数退出系统()
    Dim intReturn As Integer
    intReturn = MsgBox("真的退出系统码？", vbYesNo + vbQuestion, "提示")
    If intReturn = vbYes Then Application.Quit
End Sub
```

执行以上代码，将显示如图 5-10 所示对话框。单击【是】按钮，将执行 Quit 方法退出 Excel。单击【否】按钮，将返回应用程序。

提示：MsgBox 函数也可写成语句的形式，这样，程序就不知道用户单击了对话框中的哪个按钮。这种形式常用来显示提示信息。

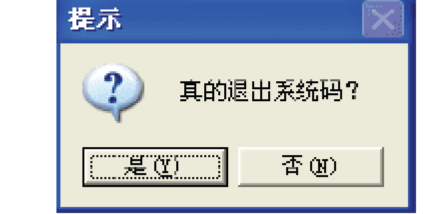


图 5-10　提示对话框

5.3　分支程序

分支程序就是根据不同的条件进行判断，选择要执行的代码段。例如，在工资管理系统中调整员工工资时，若职称为工程师则上调 100，工龄大于 20 年的上调 50。这种情况就

是分支结构，当满足条件时执行一部分代码，不满足条件时执行另一部分代码。

VBA 中的条件判断语句有 If 语句和 Select Case 语句两种。

5.3.1 单分支语句——If...Then

最常用的分支语句就是 If…Then 语句，用 If...Then 结构可有条件地执行一个或多个语句。该语句有以下两种语法形式：

- 单行结构条件语句；
- 块结构条件语句。

1. 单行结构条件语句

单行结构条件语句是最基本的条件语句，其语法为：

```
If 逻辑表达式 Then 语句
```

该语句的功能是：若逻辑表达式的值是 True，则执行 Then 后的语句，执行完成后执行 If 语句的下一句；若逻辑表达式的值是 False，则不执行 Then 后的语句，而执行 If 语句的下一条语句。其流程图如图 5-11 所示。

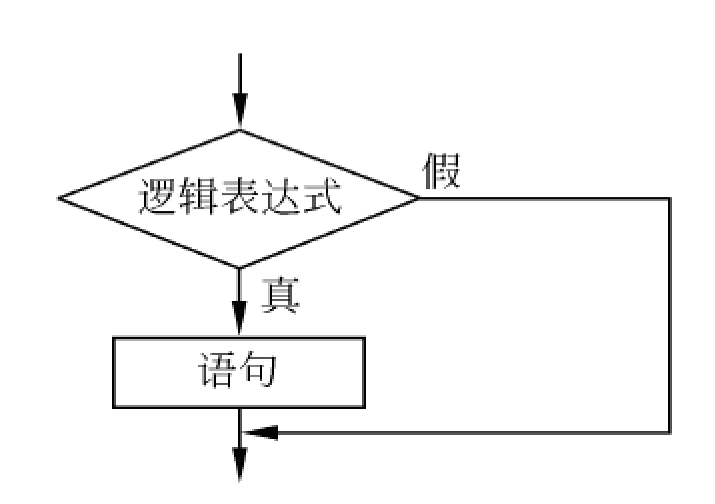


图 5-11 If...Then 语句流程图

技巧： “逻辑表达式”也可以是任何计算数值的表达式，VBA 将这个值解释为 True 或 False：为 0 的数值为 False，而任何非零数值都被看作 True。

例如，以下代码用来调整职称为工程师的员工的工资。

```
If 职称 = "工程师" Then 工资 = 工资 + 100
```

执行以上语句，如果职称为工程师，则工资增加 100，否则不进行任何操作。

2. 块结构条件语句

单行结构条件语句中，满足条件时只执行一条语句，若有多行语句需要执行，则需使用块结构条件语句。其语法如下：

```
If 逻辑表达式 Then
   语句序列 1
   语句序列 2
```

```
  ……
End If
```

块结构条件语句的作用与单行结构条件语句的功能相同。要注意的是 If...Then 的单行格式不用 End If 语句，而块语句则必须在条件语句的结束处有 End If。

例如，以下代码用来调整职称为工程师的员工的工资、岗位津贴的值。

```
If 职称 = "工程师" Then
    工资 = 工资 + 100
    岗位津贴 = 岗位津贴 + 10
End If
```

5.3.2　二分支语句——If ... Then ... Else

If…Then 语句中，当逻辑表达式的值为 False 时，不执行任何语句，若要求在逻辑表达式的值为 False 时需要执行另一段代码，可使用 If ... Then … Else 语句，其语法格式如下：

```
If 逻辑表达式 Then
   语句序列 1
Else
   语句序列 2
End If
```

在以上语句结构中，VBA 首先判断逻辑表达式的值，若其值为 True，则执行语句序列 1，执行完毕后再执行 End If 后的语句。若逻辑表达式的值为 False，则执行语句序列 2。其流程图如图 5-12 所示。

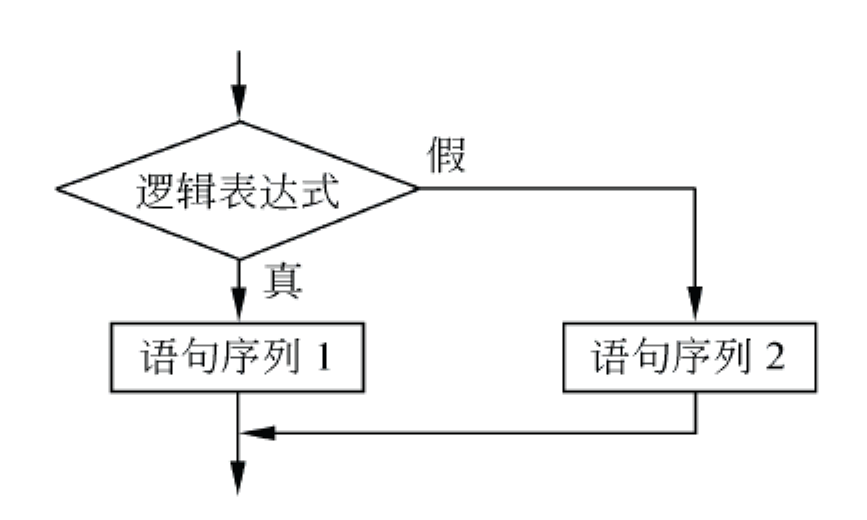


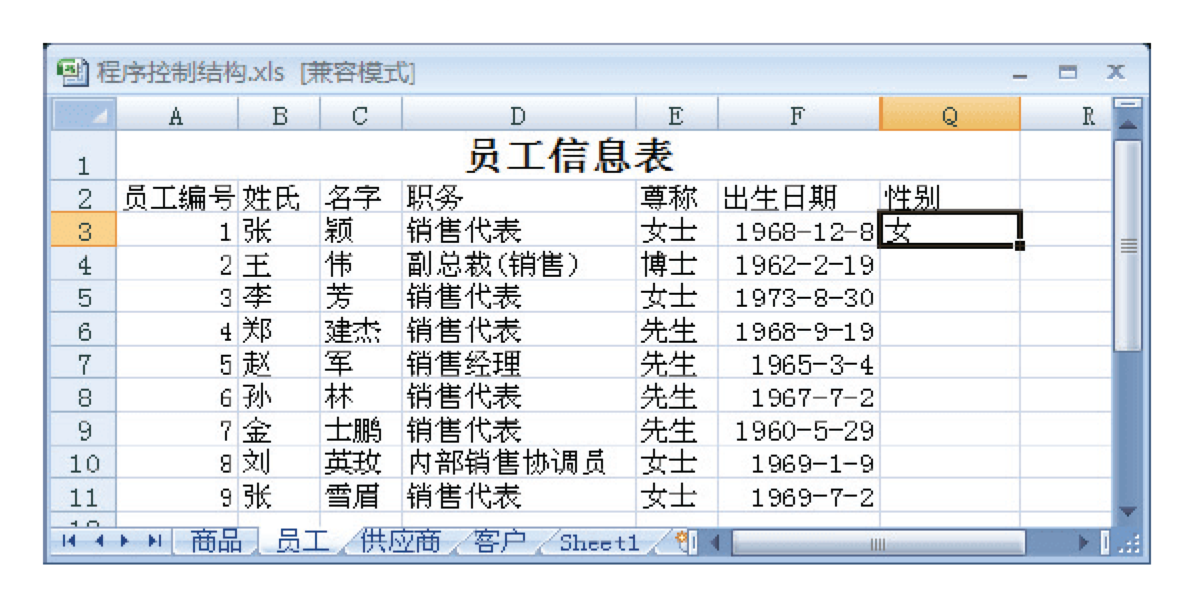
图 5-12　If…Then…Else 语句流程图

例如，以下代码根据工作表员工中的 E 列的值，在 Q 列中填入对应的性别。

```
Sub 填写性别()
    Worksheets("员工").Activate
    With Worksheets("员工")
        If .Cells(3, 5) = "女士" Then
            .Cells(3, 17) = "女"
        Else
            .Cells(3, 17) = "男"
```

```
        End If
    End With
End Sub
```

执行以上代码，员工工作表中的 Q 列将自动填入男或女，如图 5-13 所示。



	A	B	C	D	E	F	Q
1	员工信息表						
2	员工编号	姓氏	名字	职务	尊称	出生日期	性别
3	1	张	颖	销售代表	女士	1968-12-8	女
4	2	王	伟	副总裁(销售)	博士	1962-2-19	
5	3	李	芳	销售代表	女士	1973-8-30	
6	4	郑	建杰	销售代表	先生	1968-9-19	
7	5	赵	军	销售经理	先生	1965-3-4	
8	6	孙	林	销售代表	先生	1967-7-2	
9	7	金	士鹏	销售代表	先生	1960-5-29	
10	8	刘	英玫	内部销售协调员	女士	1969-1-9	
11	9	张	雪眉	销售代表	女士	1969-7-2	

图 5-13　使用 If … Then … Else 语句

5.3.3　多分支语句——If … Then … ElseIf

在很多程序中，可能需要判断多个条件，再根据不同的条件执行不同的语句。这时可使用 If…Then…ElseIf 语句来选择多个条件语句中的一部分进行执行。其语法格式如下：

```
If 逻辑表达式 1 Then
    语句序列 1
ElseIf 逻辑表达式 2 Then
    语句序列 2.
    ……
Else
    语句序列 n
End If
```

在以上结构中，VBA 首先判断逻辑表达式 1 的值。如果为 False，再判断逻辑表达式 2 的值，依此类推，当找到一个为 True 的条件，就会执行相应的语句块，执行完指定的语句块后，再执行 End If 后面的代码。其流程图如图 5-14 所示。

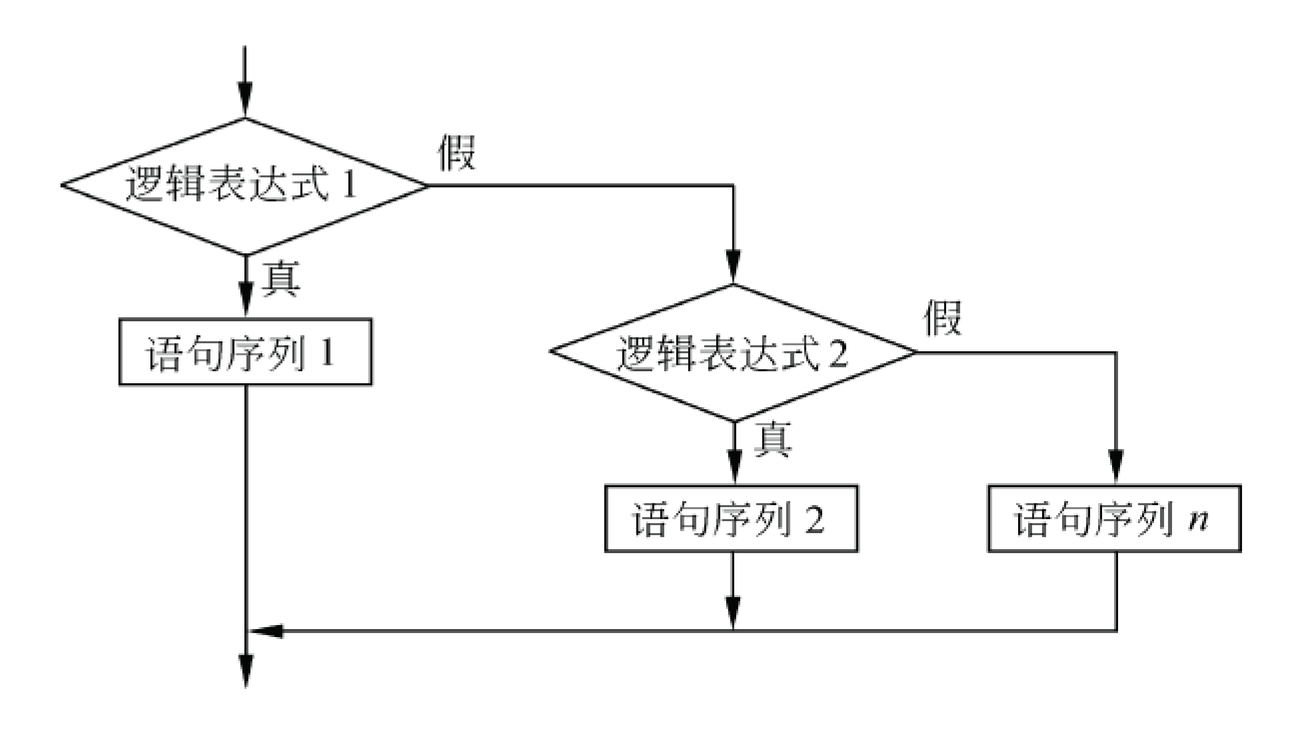


图 5-14　If … Then … ElseIf 语句

例如，以下代码演示 If…Then…ElseIf 语句的用法：

```
Sub If多分支程序()
    Dim str1 As String        '保存输入值
    Dim sPrompt As String     '提示信息
    Dim sTitle As String      '标题
    Dim sDefault As String    '默认值
    Dim sTemp As String
    sPrompt = "请输入员工的职称：" & vbNewLine & _
        "1：高级工程师" & vbNewLine & _
        "2：工程师" & vbNewLine & _
        "3：助理工程师" & vbNewLine & _
        "4：技术员" & vbNewLine & _
        "输入其他值，则无职称"
    sTitle = "输入职称"
    sDefault = "2"
    str1 = InputBox(sPrompt, sTitle, sDefault)
    If str1 = "" Then Exit Sub  '用户单击【取消】按钮，退出程序

    If str1 = "1" Then
        sTemp = "高级工程师"
    ElseIf str1 = "2" Then
        sTemp = "工程师"
    ElseIf str1 = "3" Then
        sTemp = "助理工程师"
    ElseIf str1 = "4" Then
        sTemp = "技术员"
    Else
        sTemp = "其他"
    End If
    Worksheets("sheet1").Range("A1") = sTemp
End Sub
```

执行以上代码，首先将显示如图 5-15 所示的输入对话框，用户输入一个数值后，单击【确定】按钮，即可根据输入的不同数值在工作表 Sheet1 的单元格 A1 中填入对应的职称。

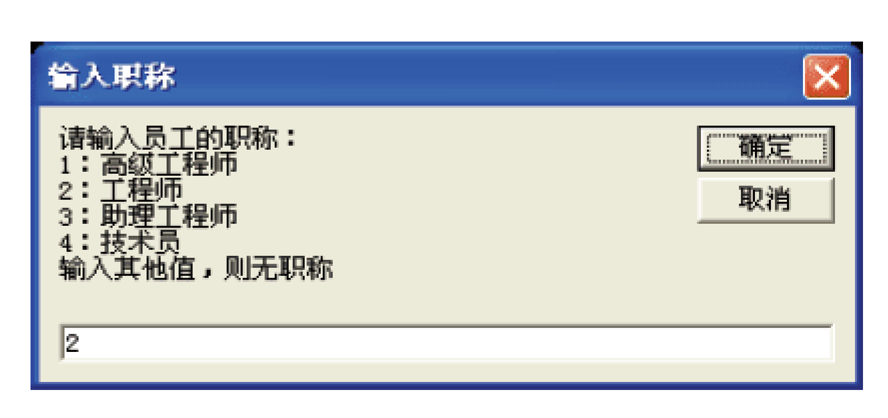


图 5-15　输入框

5.3.4　多分支语句——Select Case

在 If ... Then 分支语句中，总是可以添加更多的 ElseIf 块来构造多分支语句。但是，当每个 ElseIf 都将相同的表达式比作不同的数值时，这个结构编写起来很乏味，也不易阅读。

在这种情况下，可以用多分支选择结构 Select Case 语句。

Select Case 语句的功能与 If...Then...Else 语句类似，但对多重选择的情况，Select Case 语句使代码更加易读。

Select Case 在结构的上方处理一个测试表达式并只计算一次。然后，VBA 将表达式的值与结构中的每个 Case 的值进行比较。如果相等，就执行与该 Case 相关联的语句块，执行完毕再跳转到 End Select 语句后执行。其语法格式如下：

```
Select Case 测试表达式
Case 表达式列表 1
     语句序列 1
Case 表达式列表 2
    语句序列 2
    ……
Case Else
    语句序列 n
End Select
```

其中测试表达式可以是数值型或字符型的表达式，通常是一个数值型或字符型的变量。表达式列表可以是一个或几个值的列表。如果在一个列表中有多个值，就用逗号将值隔开。每一个语句序列中含有 0 个或多个语句。如果不止一个 Case 与测试表达式相匹配，则只对第一个匹配的 Case 执行与之相关联的语句块；如果在表达式列表中没有一个值与测试表达式相匹配，则 VBA 执行 Case Else 子句（此项是可选的）中的语句。其流程图如图 5-16 所示。

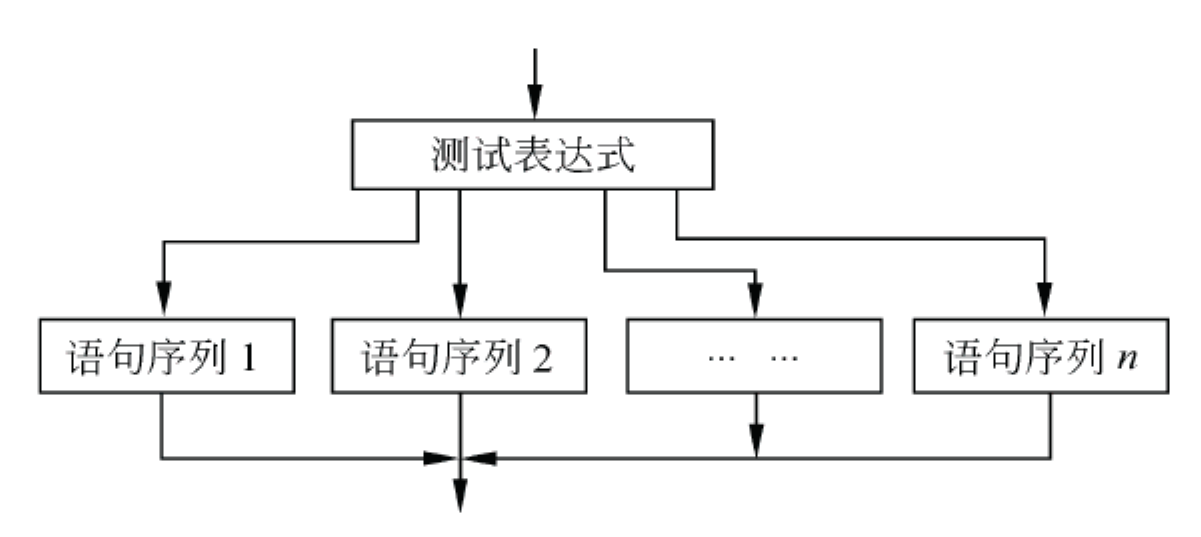


图 5-16　Select Case 语句流程图

表达式列表可以按以下几种情况进行书写。

- 表达式：这种方式用来表达一些具体的取值。例如，Case 1，3，5。
- 表达式 A To 表达式 B：这种方式用来表示一个数据范围。例如，Case 1 To 10。
- Is 比较运算符表达式。例如，Case Is<60 表示所有小于 60 的值。
- 以上 3 种情况的混合。例如，Case 0 To 60，80，Is>90。

注意：Select Case 结构每次都要在开始处计算表达式的值。If...Then...Else 结构为每个 ElseIf 语句计算不同的表达式。只有在 If 语句和每一个 ElseIf 语句计算的表达式相同时，才能用 Select Case 结构替换 If...Then...Else 结构。

例如，使用 Select Case 语句改写 5.3.3 节的 If … Then … ElseIf 语句的多分支实例，具

体的代码如下：

```
Sub Case多分支程序()
    Dim str1 As String         '保存输入值
    Dim sPrompt As String     '提示信息
    Dim sTitle As String      '标题
    Dim sDefault As String    '默认值
    Dim sTemp As String
    sPrompt = "请输入员工的职称：" & vbNewLine & _
        "1：高级工程师" & vbNewLine & _
        "2：工程师" & vbNewLine & _
        "3：助理工程师" & vbNewLine & _
        "4：技术员" & vbNewLine & _
        "输入其他值，则无职称"
    sTitle = "输入职称"
    sDefault = "2"
    str1 = InputBox(sPrompt, sTitle, sDefault)
    If str1 = "" Then Exit Sub  '用户单击【取消】按钮，退出程序

    Select Case str1
    Case "1"
        sTemp = "高级工程师"
    Case "2"
        sTemp = "工程师"
    Case "3"
        sTemp = "助理工程师"
    Case "4"
        sTemp = "技术员"
    Case Else
        sTemp = "其他"
    End Select
    Worksheets("sheet1").Range("A1") = sTemp
End Sub
```

5.4　循环程序结构

在顺序结构的程序中，每一个语句只能执行一次。在分支结构的程序中，根据逻辑表达式的值选择某一分支执行，所选分支的语句也只执行一次。然而在处理实际问题的过程中，经常要利用同一种方法对不同的数据进行重复处理，这些相同的操作可以通过重复执行同一程序段来实现。这种重复执行具有特定功能程序段的程序就称为循环程序。

在 VBA 中，通过循环控制语句来实现循环结构。

5.4.1　了解循环程序

在 5.3.2 节中的例子根据 Excel 工作表中尊称列的值，自动在性别列填充相应的性别。该例的代码只能对指定的行进行操作，若需要对当前工作表的所有行进行处理，则可使用

循环程序来进行处理。将该例的代码修改为如下形式：

```
Sub 循环填写性别()
    Worksheets("员工").Activate
    With Worksheets("员工")
    For i = 3 To 11
        If .Cells(i, 5) = "女士" Then
            .Cells(i, 17) = "女"
        Else
            .Cells(i, 17) = "男"
        End If
    Next
    End With
End Sub
```

执行以上代码，工员工作表中性别列中的每一行都将填写相应的性别数据，如图 5-17 所示。

员工信息表

员工编号	姓氏	名字	职务	尊称	出生日期	性别
1	张	颖	销售代表	女士	1968-12-8	女
2	王	伟	副总裁(销售)	博士	1962-2-19	男
3	李	芳	销售代表	女士	1973-8-30	女
4	郑	建杰	销售代表	先生	1968-9-19	男
5	赵	军	销售经理	先生	1965-3-4	男
6	孙	林	销售代表	先生	1967-7-2	男
7	金	士鹏	销售代表	先生	1960-5-29	男
8	刘	英玫	内部销售协调员	女士	1969-1-9	女
9	张	雪眉	销售代表	女士	1969-7-2	女

图 5-17　循环处理

提示：有关循环语句的相关语法，将在本节后面进行介绍。

5.4.2　For...Next 语句

在上面的代码中，使用了 For…Next 语句来循环处理工作表中的每一行数据。

For…Next 语句以指定次数来重复执行循环体。For 循环的语法格式如下：

```
For 循环变量=初始值 To 终值 [Step 步长值]
    语句序列 1
     [Exit For]
    [语句序列 2]
Next [循环变量]
```

For 循环使用一个计数器变量，每执行一次循环，计数器变量的值就会按设置的步长值增加或者减少。在 For 循环中可以使用 Exit For 语句随时退出该循环。

步长值可正可负，如果步长值为正，则初始值必须小于等于终值，才执行循环体；否则退出循环。如果步长值为负，则初始值必须大于等于终值，这样才能执行循环体。如果没有关键字 Step，则步长值默认为 1。For…Next 循环结构的流程图如图 5-18 所示。

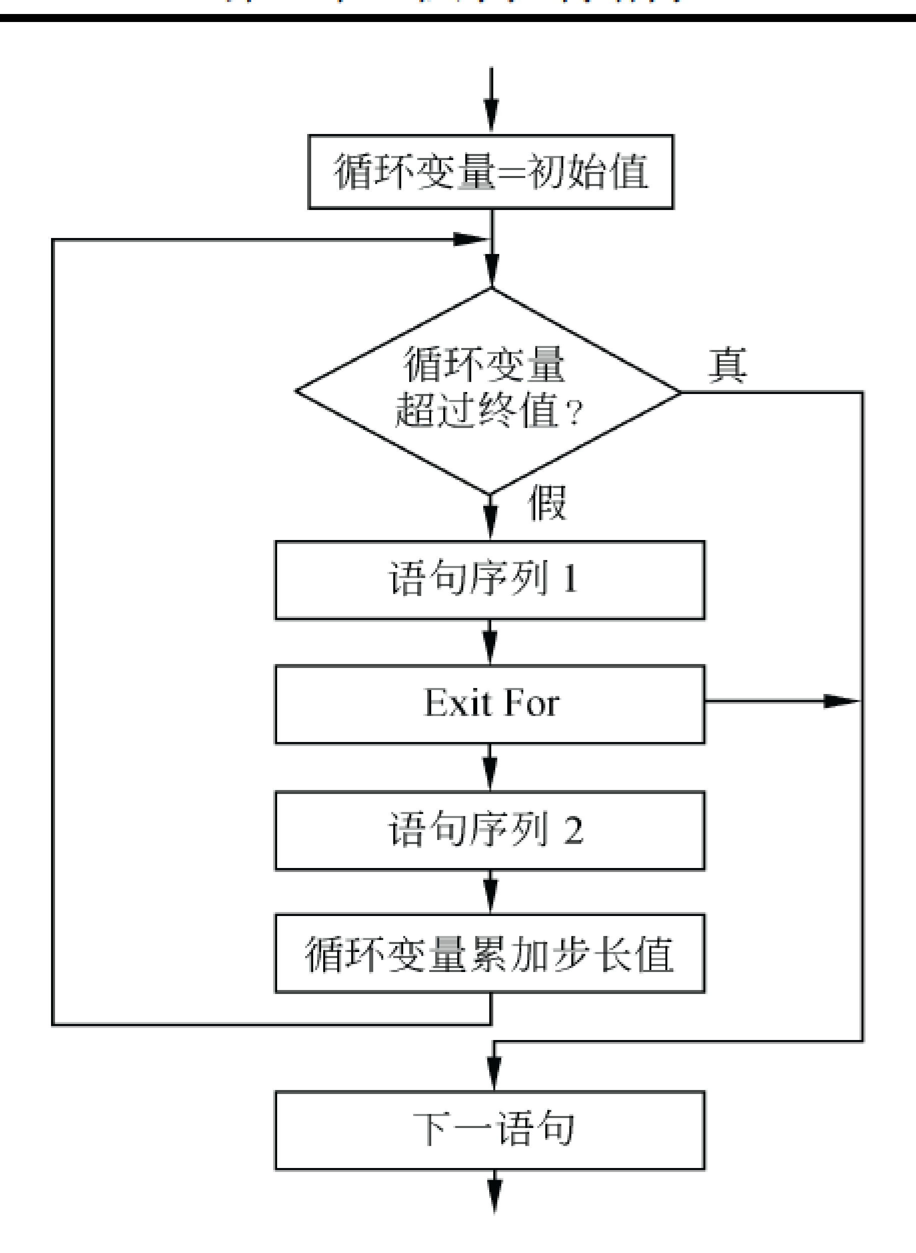


图 5-18　For …Next 流程图

For 循环一般都可计算出循环体的执行次数，计算公式如下：

```
循环次数=[(终值-初值)/步长值]+1
```

这里的中括号表示对运算结果取整，即取得除法运算商的整数部分。

例如，上例中的循环语句如下：

```
For i = 3 To 11
```

可计算出该循环的执行次数为：[(11–3)/1]+1=9 次。

若将上例代码改写为如下形式：

```
Sub 隔行填写性别()
    Worksheets("员工").Activate
    With Worksheets("员工")
    For i = 3 To 11 Step 2
        If .Cells(i, 5) = "女士" Then
            .Cells(i, 17) = "女"
        Else
            .Cells(i, 17) = "男"
        End If
    Next
    End With
End Sub
```

其循环次数为：[(11–3)/2]+1=5 次。

在工作表员工中删除性别列中各数据，执行以上代码后，将在工作表的奇数行中填入性别数据，如图 5-19 所示。由图可以看出，循环共执行了 5 次，分别在 3、5、7、9、11 各行填充了数据。

程序控制结构.xls [兼容模式]

	A	B	C	D	E	F	Q	R
1	员工信息表							
2	员工编号	姓氏	名字	职务	尊称	出生日期	性别	
3	1	张	颖	销售代表	女士	1968-12-8	女	
4	2	王	伟	副总裁(销售)	博士	1962-2-19		
5	3	李	芳	销售代表	女士	1973-8-30	女	
6	4	郑	建杰	销售代表	先生	1968-9-19		
7	5	赵	军	销售经理	先生	1965-3-4	男	
8	6	孙	林	销售代表	先生	1967-7-2		
9	7	金	士鹏	销售代表	先生	1960-5-29	男	
10	8	刘	英玫	内部销售协调员	女士	1969-1-9		
11	9	张	雪眉	销售代表	女士	1969-7-2	女	

商品 员工 供应商 客户 Sheet1

图 5-19 隔行填充数据

5.4.3 Do...Loop 语句

用 Do 循环重复执行一语句块，且重复次数不定。Do...Loop 语句有 4 种演变形式，但每种都需要计算条件表达式的值，以决定是否继续执行。在 Do 循环中可以使用 Exit Do 语句中途退出该循环。

1. 先测试循环条件的Do…Loop语句

先测试循环条件的 Do…Loop 语句语法形式如下：

```
Do While 逻辑表达式
    语句序列 1
    [Exit Do]
    [语句序列 2]
Loop
```

当 VBA 执行这个 Do 循环时，首先判断逻辑表达式，如果为 False（或 0），则跳过所有语句，执行 Loop 的下一条语句，如果为 True（或非零），则执行循环体，当执行到 Loop 语句后，又跳回到 Do While 语句再次判断条件。在循环体中如果包含有 Exit Do 语句，当执行到该语句时，则马上跳出循环，执行 Loop 的下一条语句。其流程图如图 5-20 所示。

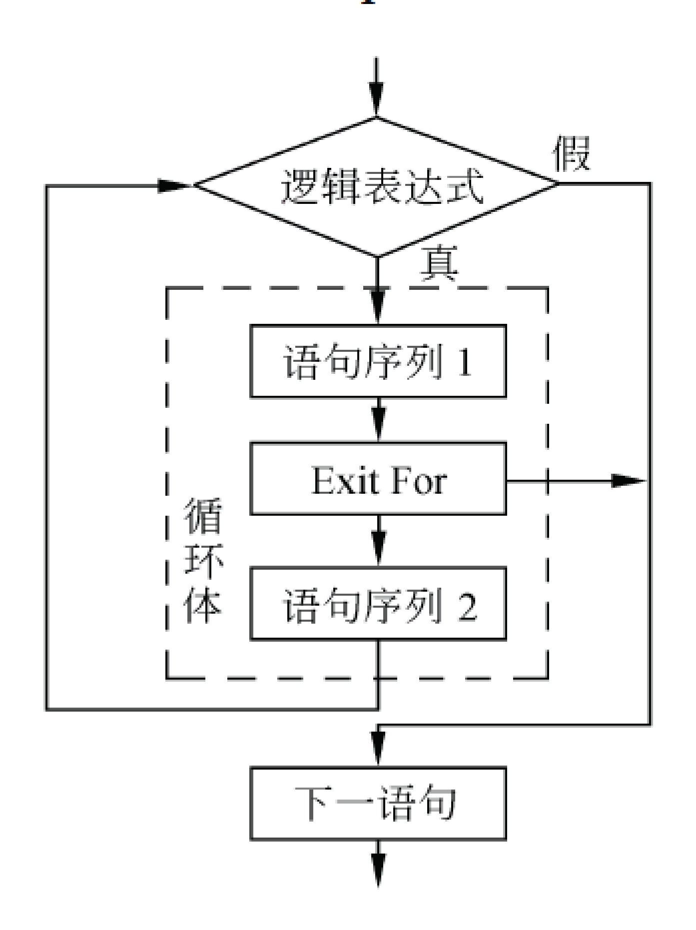


图 5-20 Do While … Loop 流程图

这种形式的循环体可能执行 0 次或多次。只要条件表达式为 True 或非零，循环就会重复执行多次。如果逻辑表达式最初就为 False，则不会执行循环语句。

2. 后测试循环条件的Do … Loop语句

Do...Loop 语句的另一种演变形式是先执行循环体中的语句，然后再进行条件判断。这种形式的循环体至少执行一次，语法格式如下：

```
Do
    语句序列 1
    [Exit Do]
    [语句序列 2]
Loop While 逻辑表达式
```

其流程图如图 5-21 所示。

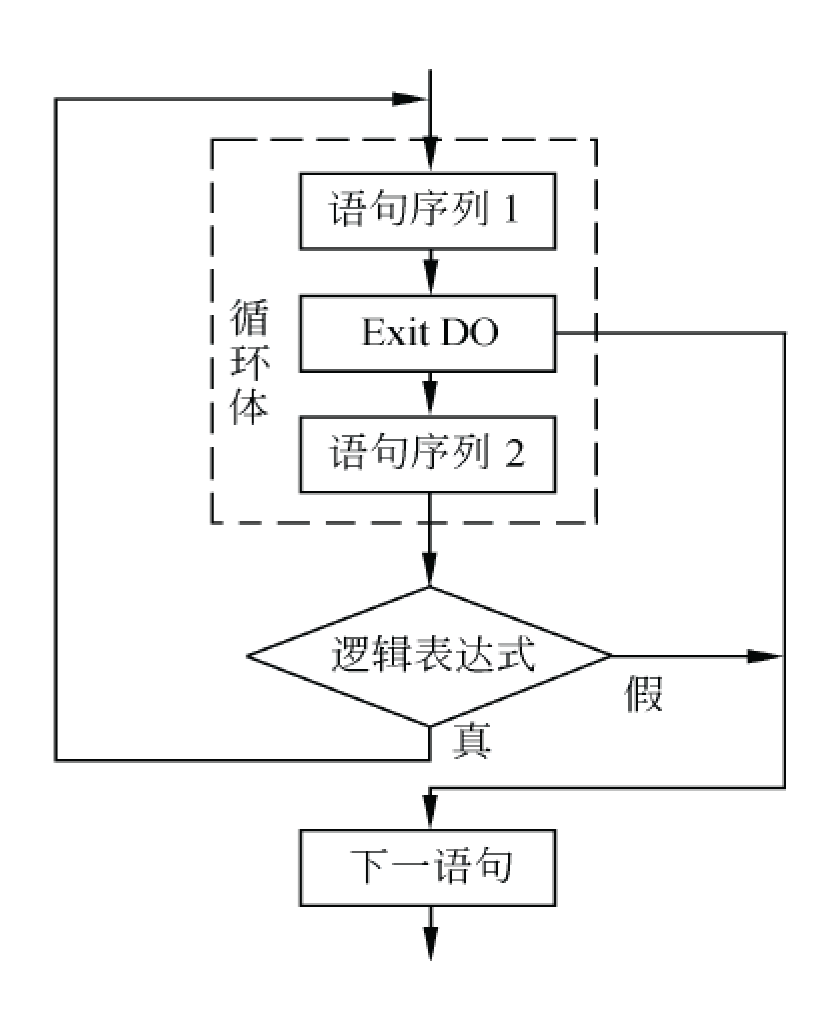


图 5-21　Do … Loop While 流程图

3. 先测试结束条件的Do … Loop语句

先测试结束条件的 Do…Loop 语句的语法形式如下：

```
Do Until 逻辑表达式
    语句序列 1
    [Exit Do]
    [语句序列 2]
Loop
```

这种形式与 Do While…Loop 类似，不同的是：当逻辑表达式的值为 False 时才执行循环体；否则退出循环。这种形式的循环体可能执行 0 次或多次。

4. 后测试结束条件的Do … Loop语句

后测试结束条件的 Do…Loop 语句的语法形式如下：

```
Do
    语句序列 1
     [Exit Do]
    [语句序列 2]
Loop Until 逻辑表达式
```

这种形式与 Do…Loop While 类似，不同的是：当逻辑表达式的值为 False 时才执行循环体；否则退出循环。这种形式的循环体至少能被执行一次。

使用 Do…Loop 循环修改 5.4.2 节的 For 循环，具体代码如下：

```
Sub Do 循环填写性别()
    Worksheets("员工").Activate
    With Worksheets("员工")
    i = 3     '循环变量赋初值
    Do While i <= 11
        If .Cells(i, 5) = "女士" Then
            .Cells(i, 17) = "女"
        Else
            .Cells(i, 17) = "男"
        End If
        i = i + 1     '必须在循环体中改变循环变量
    Loop
    End With
End Sub
```

在 For 循环中，执行到 Next 语句时将自动修改循环变量的值。而 Do 循环没有自动修改循环变量的功能，所以需要在循环体中添加语句，用来修改循环变量的值。若没有修改循环变量的语句，Do 循环将是一个死循环，一直不能退出循环。

5.4.4　For Each...Next 语句

For Each...Next 循环与 For...Next 循环类似，但它针对数组或对象集合中的每一个元素重复一组语句，而不是重复语句一定的次数。如果不知道一个集合有多少元素，For Each...Next 循环非常有用。For Each...Next 循环的语法如下：

```
For Each 对象元素变量 In 对象集合
    语句序列 1
     [Exit For]
    [语句序列 2]
Next 对象元素变量
```

使用 For Each...Next 时，根据对象集合的不同，有不同的限制。

- ❑ 对于集合：对象元素变量只能是 Variant 变量，或一般的对象（Object）变量，或对象浏览器中列出的对象。
- ❑ 对于数组：对象元素变量只能是 Variant 变量。

For Each...Next 不能与用户自定义类型的数组一起使用，因为 Variant 不可能包含用户自定义类型。

🔔**提示：**For Each 循环一般使用在对象模型中，相关的实例会在本书后面章节中列出。

5.4.5　循环嵌套

前面学习的都是单层控制结构。其实，可以把一个控制结构放入另一个控制结构之内（例如在 For...Next 循环中的 If...Then 块）。

循环嵌套就是在一个循环中还有一个循环，内部循环在外部循环体中。在外部循环的每次执行过程中都会触发内部循环，直到内部循环执行结束。外部循环执行了多少次，内部循环就完成多少次。

按照一般习惯，为了使判定结构和循环结构更具可读性，总是用缩排方式书写判定结构或循环的正文部分。

例如，使用循环嵌套编写冒泡排序法程序，具体代码如下：

```
Sub 循环嵌套()
    '本例使用冒泡排序法演示循环嵌套
    Dim i As Integer
    Randomize
    With Worksheets("sheet1")
    For i = 1 To 100                                   '生成 100 个随机数据
        .Cells(i, 1) = Int(Rnd * 100) + 1
    Next
    For i = 1 To 99                                    '外循环
        For j = i + 1 To 100                           '内循环
            If .Cells(i, 1) > .Cells(j, 1) Then        '比较数据大小
                t = .Cells(i, 1)
                .Cells(i, 1) = .Cells(j, 1)
                .Cells(j, 1) = t
            End If
        Next
    Next
    End With
End Sub
```

执行以上代码，首先使用单循环生成 100 个随机整数，接着使用循环嵌套对数据进行排序，得到如图 5-22 所示的排序结果。

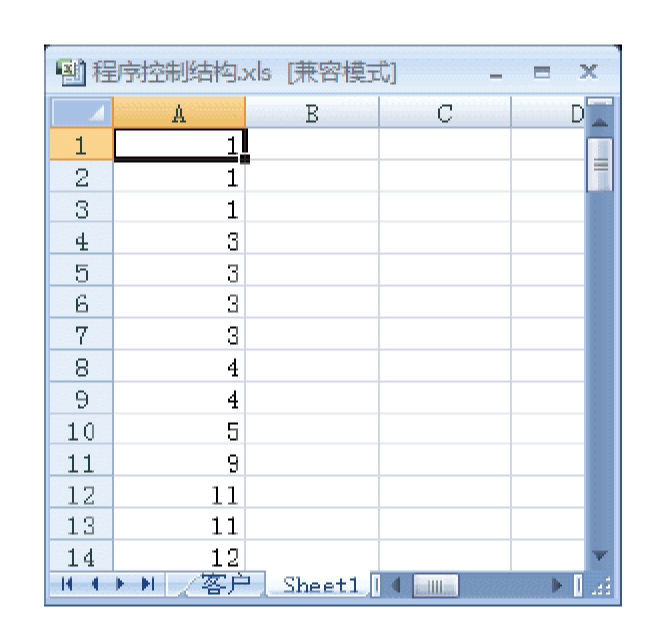


图 5-22　排序结果

本例中的外循环共循环 99 次（即一共输出九行），当 i 等于 100 时循环终止。外循环的每轮循环都会执行内循环，在外循环的每轮循环中，内循环的循环次数都不相同。因为外循环的每轮循环都会使 i 增加 1，而内循环的循环变量 j 的值也会被重新赋值为 i+1，而内循环的结束条件是 100，且内循环的每轮循环中 j 只增加 1，所以外循环每循环一次，内循环的循环次数就减少一次，即在外循环的第 1 轮循环，内循环的循环次数为 99；在外循环的第 2 轮循环，内循环的循环次数为 98；在外循环的第 3 轮循环，内循环的循环次数为 97……

提示： 在嵌套结构里的循环结构中使用 Exit 语句时，退出的只是包含该语句的当前循环结构，而不是整个嵌套结构。

第 6 章　使 用 数 组

前面各章中使用的数据都是基本数据类型，可以通过简单的变量名来访问其保存的值。除了基本数据类型外，VBA 还提供了数组类型，利用数组可以方便地组织和使用数据。例如，可用数组保存工作表中各单元格的数据。

本章介绍数组的定义和使用方法。

6.1　数 组 简 介

数组是有序的数据的集合，在其他高级程序设计语言中，数组中的所有元素都属于同一个数据类型，而在 VBA 中，一个数组中的元素可以是不同类型的数据，也可以是相同类型的数据。

6.1.1　用数组保存工作表数据

使用数组时，可以用同一名称引用多个值，并使用一个称作索引或下标的数字将它们区分开来。数组可以缩短和简化代码，使程序员能够创建高效处理多个元素的循环。

Excel 工作表中的单元格由行和列组成，在 Excel 2003 及以前版本中，每行最多可有 255 列，每张工作表最多有 65 536 行。而 Excel 2007 中，工作表中单元格的数量得到了很大的扩展，每行最多可有 16 384 列，每张工作表最多有 1 048 576 行。

在 VBA 中，如果使用变量来保存单元格中的数据时，要保存一行的数据，则需要定义多个变量，例如，用 V1 保存单元格 A1 的值，用 V2 保存单元格 B1 中的值，……在程序中，若需要分别访问这些变量中的值，则需要分别编写代码。例如，将 Excel 工作表中第 1 行各单元格的数据进行累加，可使用以下代码：

```
s=s+V1
s=s+V2
s=s+V3
   ……
s=s+Vn
```

如果要对 255（或 16 384）列单元格进行累加，则需要编写 255（或 16 384）行代码。显然，这样的程序代码不是开发人员希望使用的。

这时，如果使用数组来保存单元格中的值，则可以使 V(1)、V(2)、…、V(n)来表示，也可使用 V(1)保存单元格 A1 的值，用 V(2)保存单元格 B1 中的值，……

这样，在程序中要累加各单元格中的值时，可使用以下的循环语句：

```
For i = 1 To 255
    s = s + V(i)
Next
```

以上代码中，为了兼容 Excel 2003 及以前版本，设置累加 255 列单元格的和。

由以上代码可以看出，只需要 3 行语句即可对 255 列（甚至更多）的数据进行累加。由此可以看出，使用数组可方便地对循环语句进行处理，简化程序代码。

6.1.2 数组的维数

在 6.1.1 节示例中的数组 V，在括号中使用了一个索引，因此称为“一维”。使用多个索引或下标的数组称为“多维”。

提示：“维”是一个方向，可以在此方向上改变数组元素的规范。保存月内每日总销量的数组有一个维（当月日期）。保存每个部门的月内每日总销量的数组有两个维（部门编号和当月日期）。数组的维数称为数组的“秩”。

可以通过为数组的每一维提供“索引”或“下标”来指定数组元素。在每一维中，元素都按照从小到大的索引顺序连续排列。

1. 一维数组

很多数组只有一维，例如，统计各年龄的人数时，可定义一个数组，每个数组元素表示一个年龄段的人数（如 iAgeCnt(10)中保存着年龄为 10 岁的人数）。因此，这类数组只使用一个索引。下面的代码声明一个变量来保存一个“一维数组”，其中包含从 0 岁到 120 岁之间每个年龄段的人数。

```
Dim iAgeCnt(120) As Integer
```

以上代码定义的一维数组如图 6-1 所示。

0	1	2	3	4	5	6	7	8	9	10	…	119	120

图 6-1　一维数组

使用 iAgeCnt(0)可以访问数组中的第 1 个元素，使用 iAgeCnt(10)可访问数组中的第 11 个元素。

2. 二维数组

某些数组有两个维，最典型的就是 Excel 的工作表结构，由行和列构成。二维数组也称为“矩形数组”。

下面的代码声明一个变量来保存一个“二维数组”。

```
Dim aData(4, 10) As Integer
```

以上代码定义的二维数组如图 6-2 所示。

使用 aData(0,0)可访问数组中第 1 行第 1 列的元素（第 1 个元素），使用 aData(2,2)可访问数组中的第 3 行第 3 列的元素。

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	0,10
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	1,10
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9	2,10
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9	3,10
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9	4,10

图 6-2　二维数组

3．三维数组

有些数组有三个维，如 Excel 工作簿就是一个三维空间（每张工作表为一个二维数组结构，多张工作表就构成一个三维数组），这类数组使用三个索引。

例如，使用以下代码声明一个“三维数组”，第一个索引表示工作簿中的某张工作表，第二个索引表示指定工作表中的行，第三个索引表示指定工作表中的列。

```
Dim aData(3, 4, 10) As Integer
```

以上代码定义的三维数组如图 6-3 所示。

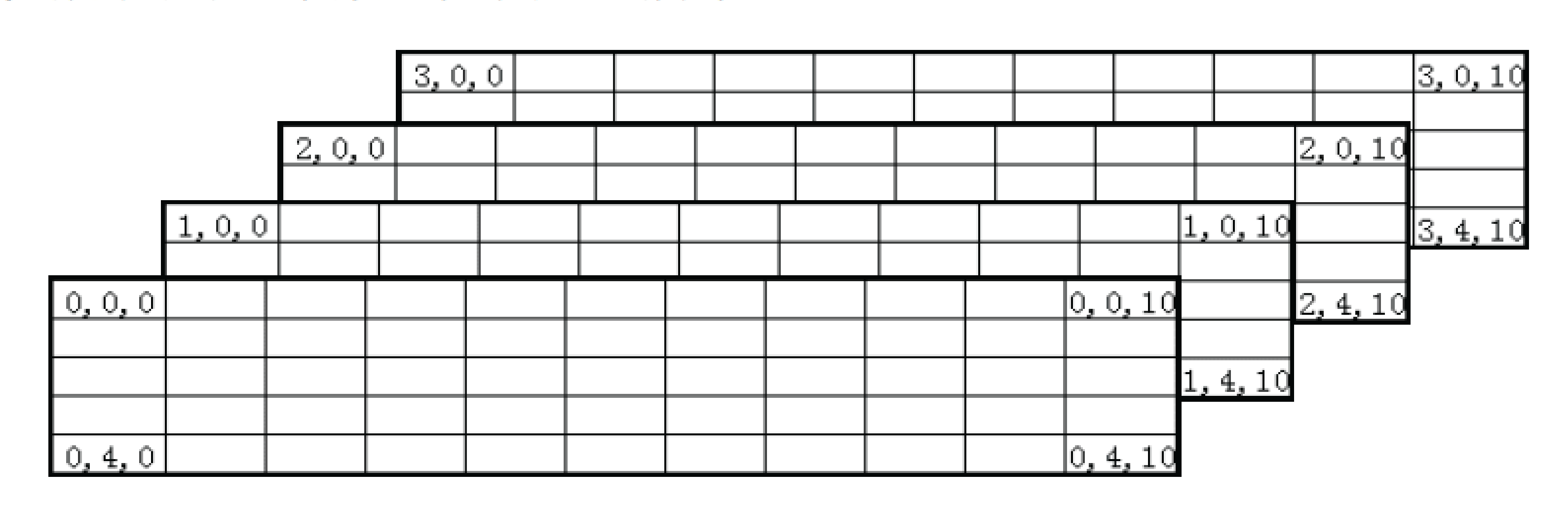


图 6-3　三维数组

使用 aData(0,0,0)可访问数组中第 1 页（第 1 个工作表）中第 1 行第 1 列的元素（第 1 个元素），使用 aData(2,2,2)可访问数组中第 3 页（第 3 个工作表）的第 3 行第 3 列的元素。

4．多维数组

多于三维的数组为多维数组。尽管 VBA 数组最大可以达到 60 维，但是超过三维的数组是非常难于理解的，所以常用的也就是一维、二维和三维数组。

注意：增加一个数组的维数时，该数组所需的总存储空间会急剧增大，因此应慎用多维数组。

6.2　声 明 数 组

可以使用 Dim 语句，用声明任何其他变量的方法来声明数组变量。在变量名后面加上

一对或几对圆括号，以指示该变量将存储数组而不是“标量”（包含单个值的变量）。

6.2.1 声明一维数组

数组的声明方式和其他的变量是一样的，可以使用 Dim、Static、Private 或 Public 语句来声明。声明标量变量（非数组）与数组变量的不同在于必须为数组指定大小。若数组的大小被指定时，则它是个固定大小数组。若程序运行时数组的大小可以被改变，则它是个动态数组。

声明一维数组的语法格式如下：

```
Dim 数组名(下界 To 上界) As 数据类型
```

与其他程序设计语言定义数组的格式不同，在 VBA 中定义数组时，下界的值可为任意整型值（可为负数）。

在声明数组时，也可只给出数组下标的上界（即可以使用的最大下标值）。而省略下标的下界，这时默认值为 0，即数组的下标从 0 开始至定义的上界，如：

```
Dim  aData(10)  As  String
```

定义了一个名为aData的数组，共有11个元素，分别为aData(0)、aData(1)、…、aData(10)。

使用以下代码可指定数组的下界：

```
Dim aData(-10 To 10) As String
```

以上代码定义了一个数组 aData，具有 21 个元素，分别为 aData(–10)、aData(–9)、aData(–8)、…、aData(0)、…、aData(9)、aData(10)。

在定义数组时，需要注意以下几点：

- 数组的名称必须符合标识符的规则，并尽可能为数组定义有一定意义的名称。
- 在同一过程中，数组名不能与变量名相同；否则会出错。
- 在 VBA 中定义数组时，要求其下标必须为常数，不能是变量或表达式，例如，以下代码在执行时将会报错，如图 6-4 所示。

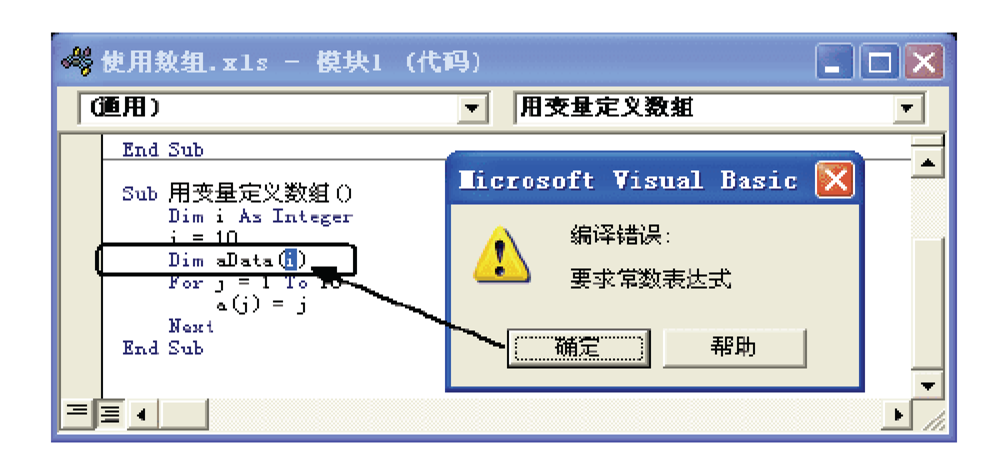


图 6-4　用变量定义数组

```
Sub 用变量定义数组()
    Dim i As Integer
    i = 10
    Dim aData(i)
    For j = 1 To 10
```

```
        a(j) = j
    Next
End Sub
```

6.2.2　声明多维数组

定义多维数组的语法格式如下：

```
Dim 数组名([第 1 维下界 To] 第 1 维上界, [第 2 维下界 To] 第 2 维上界, … , [第 n 维下界
To] 第 n 维上界)  As 数据类型
```

使用以上格式可声明二维数组、三维数组、多维数组等。

例如，使用以下代码可声明一个二维数组，用来保存 Excel 工作表中的值：

```
Dim aData(1 To 10, 1 To 255)
```

以上代码定义的二维数组可保存 Excel 工作表中的 10 行数据，每行可保存 255 列（共 2550 个单元格的值）。为了使数组元素保存不同类型的值，在声明数组时，不能定义数组的数据类型。

定义多维数组的格式与二维数组类似，每一维都使用逗号隔开即可。需要注意的是，定义大数据量的数组将占用很大的内存空间，特别是定义多维数组时要考虑这一点。例如：

```
Dim aData(1 To 65536) As Integer
```

定义一个数组，具有 65 536 个元素，相当于定义 65 536 个变量。

```
Dim aData(1 To 65536, 1 To 100) As Integer
```

定义一个二维数组，具有 6 553 600（65 536×100）个元素，而

```
Dim aData(1 To 65536, 1 To 100, 1 To 100) As Integer
```

定义一个三维数组，具有 655 360 000（65 536×100×100）个元素。

由此可见，数组每增加一维，数组元素就会成几何级数的增加。若定义的数组维数过多，可导致程序很快将内存用完。

6.2.3　设置数组默认下界

在默认情况下，在程序中声明数组时，如果不指定数组维数的下界，则 VBA 使用默认下界 0。

例如：

```
Dim aData(10)
```

以上代码将定义 11 个元素，从 aData(0)开始，至 aData(10)为止。

如果希望下标从 1 开始，可以通过 Option Base 语句来设置。该语句必须在模块级别中使用，其语法格式如下：

```
Option Base {0 | 1}
```

由于下界的默认设置是 0，因此无需使用 Option Base 语句。如果使用该语句，则必须写在模块的所有过程之前。一个模块中只能出现一次 Option Base，且必须位于带维数的数组声明之前。

Option Base 语句只影响位于包含该语句的模块中的数组下界。

例如，以下代码查看各数组下标的下界为 1。

```
Option base 1                          '将默认的数组下标设为 1
Dim Lower
Dim MyArray(20), TwoDArray(3, 4)       '声明数组变量
Dim ZeroArray(0 To 5)                  '取代默认的下标
'使用 LBound 函数来测试数组的下界
Lower = LBound(MyArray)                '返回 1
Lower = LBound(TwoDArray, 2)           '返回 1
Lower = LBound(ZeroArray)              '返回 0
```

6.3 初始化数组

数组具有多个元素，可保存多个数据。与单个变量的初始化不同，对数据进行初始化时，一般要进行批量赋值。数组初始化可有多种方法，本节介绍几种常用的数组初始化方法。

6.3.1 使用循环语句初始化数组

使用循环语句初始化数组是最常用的一种方法。利用循环语句可反复执行的特点，可快速对数组进行初始化。

例如：

```
Sub 使用循环初始化数组()
    Dim a(1 To 10) As Integer
    For i = 1 To 10
        a(i) = 0
    Next i
End Sub
```

以上代码对数组 a 中每个元素赋初值为 0。

6.3.2 使用 Array 函数初始化数组

Array 函数可返回一个包含数组的 Variant。其语法格式如下：

```
Array(arglist)
```

参数 arglist 是一个用逗号隔开的值表，这些值用于给 Variant 所包含的数组的各元素赋值。如果不提供参数，则创建一个长度为 0 的数组。

使用 Array 函数创建的数组的下界由 Option Base 语句指定的下界所决定。

注意：没有作为数组声明的 Variant 也可以表示数组。除了长度固定的字符串以及用户定义类型之外，Variant 变量可以表示任何类型的数组。尽管一个包含数组的 Variant 和一个元素为 Variant 类型的数组在概念上有所不同，但对数组元素的访问方式是相同的。

使用 Array 函数初始化数组的示例代码如下：

```
Sub 使用 Array 函数初始化数组()
    Dim a As Variant, b As Variant
    a = Array(1, 3, 5, 7, 9)
    b = Array("A", "B", "C", "D")
End Sub
```

6.3.3　用数组值初始化数组

在 VBA 中，还可以直接将一个数组的值赋值给另一个数组，以达到初始化数组的目的。例如，以下的代码：

```
Sub 用已有数组初始化数组()
    Dim a(5), b()
    For i = 0 To 5
        a(i) = i
    Next
    b = a
End Sub
```

使用这种方式初始化数组时，需要注意以下几点：

- 赋值号两边的数据类型必须一致；
- 如果赋值号左边是一个动态数组，则赋值时系统自动将动态数组 ReDim 成右边相同大小的数组；
- 如果赋值号左边是一个大小固定的数组，则数组赋值出错。

6.4　动　态　数　组

在很多情况下，数组的长度事先是无法预测的，而且有时可能需要在程序中改变数组的长度以适应新的情况，因此出现了动态数组。

6.4.1　声明动态数组

动态数组是指在程序运行时，数组元素大小可以改变的数组，当程序没有运行时，动

态数组不占据内存，因此可以把这部分内存用于其他操作。

大小不可改变的数组称为静态数组。定义静态数组时，其下标的下界和上界只能由常量来进行设置，而动态数组可使用变量来设置下标。

定义动态数组一般分两个步骤：

（1）在用户窗体、模块或过程中使用 Dim 或 Public 声明一个没有下标的数组（不能省略括号）。

（2）在过程中用 ReDim 语句重定义该数组的大小。

ReDim 语句的作用是重新指出数组的大小。它是在程序执行到 ReDim 语句时才分配存储空间。其语法格式如下：

```
ReDim [Preserve] 数组名(下标) [As 数据类型]
```

说明：

- 下标可以是常量，也可以是具有确定值的变量。
- 语句中各参量的含义与用 Dim 定义数组的语句相同。
- ReDim 语句只能用于动态数组，它可以改变每一维的大小，但不能改变维数。
- 当程序编译时，ReDim 语句中的所有数组均被说明为动态数组。在程序运行中，当执行到 ReDim 语句时，就把新的上下界重新分配给数组，数组元素的值将被初始化，所有的数值元素的值被置为 0，字符串元素被置为空字符串。
- ReDim 语句可以同 Dim 语句一样定义数组。在同一程序中，ReDim 语句还可以多次使用。在用 ReDim 语句重新定义数组之前，可以使用 Erase 语句将原来的数组删除。

在默认情况下，使用 ReDim 语句重定义数组的维数和大小时，数组中原来保存的值将全部消失。若要保存数组中原有的值，需要使用 Preserve 关键字，这样，当改变原有数组最后一维的大小时，可以保持数组中已有的数据。

注意：如果使用了 Preserve 关键字，就只能重定义数组最后一维的大小，并不能改变维数的数目。

例如：以下代码由用户输入一个数值，设置数组下标的上界，然后要求用户输入每个元素的值，并将用户输入的值输出到【立即窗口】列表框中。

```
Dim reData() As Integer
Sub 动态数组()
    Dim i As Integer, j As Integer
    i = Val(InputBox("请输入数组的上界", "定义动态数组", 5))
    ReDim reData(i)
    For j = 1 To i
        reData(j) = InputBox("请输入数组的第" & j & "个元素的值")
    Next
    For j = 1 To i
        Debug.Print reData(j)
    Next
End Sub
```

以上代码在模块的声明部分定义了一个动态数组 Darray，然后在“动态数组”过程中使用变量 i 重定义该动态数据的大小（i 的值由用户输入）。

执行以上代码，将首先打开如图 6-5 所示的对话框，让用户设置数组的上界。按用户输入的值对数组进行重设大小后，接着将打开输入对话框，要求用户输入每一个元素的值。最后在【立即窗口】列表框中输出数组中每个元素的值，如图 6-6 所示。

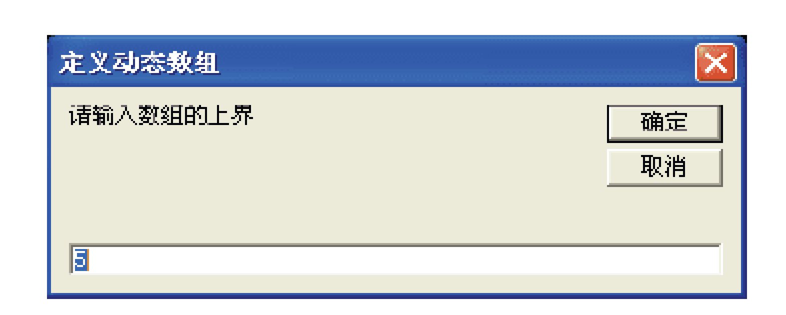


图 6-5　输入数组的上界

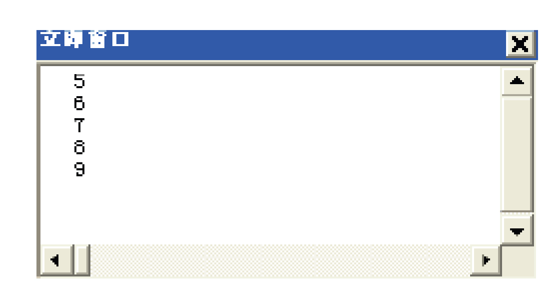


图 6-6　输出数组中的数据

6.4.2　数组的清除和重定义

对于静态数组，定义数组后，其大小不能改变。若需要清除数组的内容或对数组重新定义，可以使用 Erase 语句来实现。

Erase 语句重新初始化大小固定的数组的元素，以及释放动态数组的存储空间。其语法格式如下：

```
Erase arraylist
```

所需的 arraylist 参数是一个或多个用逗号隔开的需要清除的数组变量。

注意：在 Erase 语句中，只给出要刷新的数组名，不带括号和下标。

Erase 根据是固定大小（常规的）数组还是动态数组，来采取完全不同的行为。Erase 无需为固定大小的数组恢复内存。

Erase 释放动态数组所使用的内存。在下次引用该动态数组之前，程序必须使用 ReDim 语句来重新定义该数组变量的维数。

例如，以下代码演示 Erase 语句的功能。

```
Sub 清除数组()
    Dim aData(10) As Integer, str1 As String
    str1 = "原数组中的数据: " & vbNewLine
    For i = 0 To 10
        aData(i) = i
        str1 = str1 & "aData(" & i & ")=" & aData(i) & "  "
    Next
    Erase aData '删除原数组
    str1 = str1 & vbNewLine & "使用 Erase 命令清除数组 aData: " & vbNewLine
    For i = 0 To 10
        str1 = str1 & "aData(" & i & ")=" & aData(i) & "  "
    Next
    MsgBox str1
End Sub
```

运行以上程序，首先定义数组 aData，为该数组赋初值，并输出数组各元素的值。接着执行 Erase 语句清除数组 aData 中的值，再输出数组元素的值。最后将显示如图 6-7 所示的对话框。

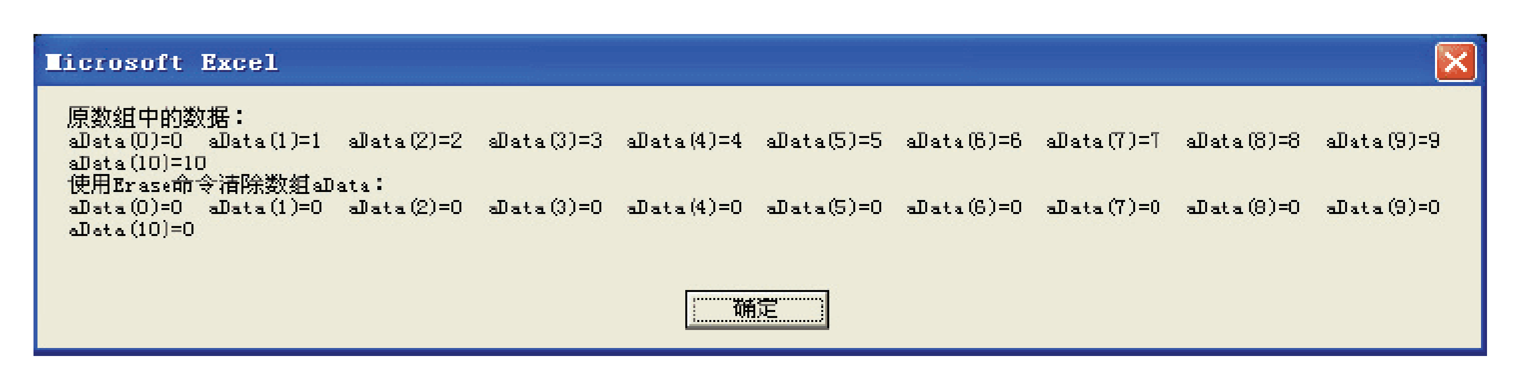


图 6-7 程序运行结果

6.5 操作数组的函数

定义数组后，在程序中可以与使用其他变量相同的操作来使用数组元素。数组是一种特殊的数据结构，因此，VBA 提供了操作数组的函数。

6.5.1 判断数组

使用 IsArray 函数可检查指定的变量是否为一个数组，如果指定变量是一个数组，返回值为 True；否则返回 False。该函数的语法格式如下：

```
IsArray(varname)
```

参数 varname 是一个指定变量的标识符。

对于包含数组的 variant 表达式来说，IsArray 尤为有用。

例如：

```
Dim aData(5) As Integer, aData1, bCheck     ' 声明数组变量
aData1= Array(1, 2, 3)                      ' 初始化数组
bCheck = IsArray(aData)                     ' 返回 True
bCheck = IsArray(aData1)                    ' 返回 True
```

6.5.2 查询数组的下标范围

使用 LBound 函数和 UBound 函数获得数组下标的下界和上界。其语法格式为：

```
LBound(数组名[, 维数])
UBound(数组名[, 维数])
```

其中“维数”为 1 表示第一维，2 表示第二维，依此类推。如果省略该参数，表示返回第一维的下标下界或上界。

例如，使用以下代码定义数组：

```
Sub 获取数组下界范围()
    Dim aData(-100 To 100, 5 To 15, -3 To 4)
    Dim str1 As String
    str1 = "数组各维的下界为: " & vbNewLine
    str1 = str1 & "第 1 维: " & LBound(aData, 1) & vbNewLine
    str1 = str1 & "第 2 维: " & LBound(aData, 2) & vbNewLine
    str1 = str1 & "第 3 维: " & LBound(aData, 3) & vbNewLine
    str1 = str1 & vbNewLine & "数组各维的上界为: " & vbNewLine
    str1 = str1 & "第 1 维: " & UBound(aData, 1) & vbNewLine
    str1 = str1 & "第 2 维: " & UBound(aData, 2) & vbNewLine
    str1 = str1 & "第 3 维: " & UBound(aData, 3) & vbNewLine
    MsgBox str1
End Sub
```

运行以上代码，将显示如图 6-8 所示的对话框，分别显示出数组各维的范围。



图 6-8　数组下标范围

6.6　数组使用实例

本章前面的内容介绍了使用数组各方面的内容，本节以实例形式演示数组的使用方法。

6.6.1　数据排序

在 Excel 中可以方便地对单元格区域中的数据进行排序。本例使用 VBA 程序首先让用户输入 10 个数据，然后使用冒泡排序法对这 10 个数据进行排序。

在 VBE 模块中编写以下代码：

```
Option Base 1
Sub 数据排序()
    Dim i As Integer, j As Integer
    Dim k
    Dim s(10) As Integer
    For i = 1 To 10
        s(i) = Application.InputBox("输入第" & i & "个数据: ", "输入数组", , , , , ,
        1)
```

```
    Next
    For i = 1 To 9
        For j = i + 1 To 10
            If s(i) < s(j) Then
                t = s(i)
                s(i) = s(j)
                s(j) = t
            End If
        Next
    Next
    For Each k In s
        Debug.Print k
    Next
End Sub
```

运行上面的代码，弹出如图 6-9 所示的对话框，提示用户输入数据。循环程序要求用户输入 10 个数据。最后在【立即窗口】输出排序的结果，如图 6-10 所示。



图 6-9 【输入数组】对话框

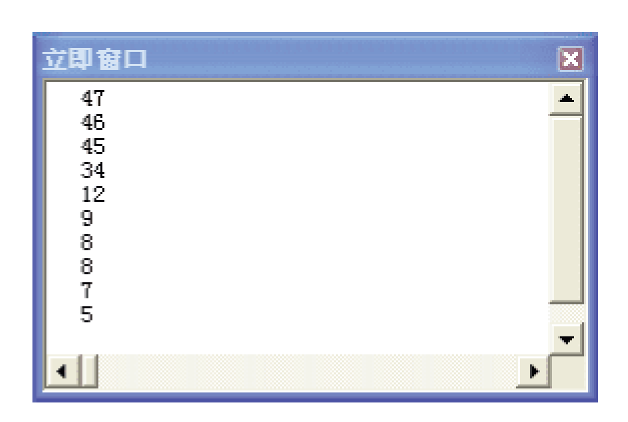


图 6-10 排序结果

6.6.2 彩票幸运号码

本例结合数组和随机函数的知识，生成指定数量的彩票幸运号码。本例生成的彩票号码每注由 7 位数构成，首先让用户输入产生的注数，再使用循环语句生成指定注数的号码。

在模块中编写以下代码：

```
Option Base 1
Sub 幸运号码()
    Dim n As Integer, i As Integer, j As Integer
    Dim l() As Integer
    n = Application.InputBox("请输入需要产生幸运号码的数量：", "幸运号码", , , , , ,
    2)
    ReDim l(n, 7) As Integer
    For i = 1 To n
        For j = 1 To 7
            Randomize
            l(i, j) = Int(10 * Rnd)
        Next
    Next
    For i = 1 To n
        For j = 1 To 7
```

```
            Debug.Print l(i, j);
        Next
        Debug.Print
    Next
End Sub
```

运行上面的宏，弹出如图 6-11 所示的对话框，提示用户输入数据。输入生成幸运号码的数量后，单击【确定】按钮将在【立即窗口】列表框中输出生成的幸运号码，如图 6-12 所示。

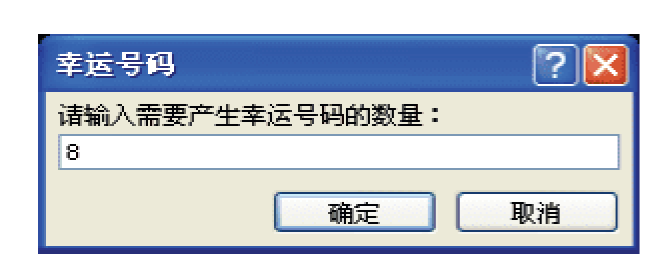


图 6-11　输入数据



图 6-12　生成幸运号码

6.6.3　用数组填充单元格区域

在 Excel 中要处理大量数据时，可使用循环从各单元格中读入数据，经过加工处理后再写回单元格区域中。这种方式比在数组中处理数据的速度要慢。因此，如果有大量的数据需要处理时，可先将数据保存到数组中，经过加工处理后，再将数组的数据填充到单元格区域。

在 Excel 工作表中，工作表是一个二维结构，由行和列组成。这种特性与二维数组类似，因此可以很方便地将工作表单元格区域与二维数组进行相互转换。通过以下语句可将单元格区域赋值给一个二维数组。

```
myarr = Range(Cells(1, 1), Cells(5, 5))
```

反过来，也可将二维数组中的值快速地赋值给一个单元格区域，如以下语句将二维数组 myarr 中的值赋值给单元格区域 Rng。

```
Rng.Value = arr
```

本例在模块中编写以下代码，用数组来处理工作表中单元格的数据。

```
Option Base 1
Sub 数组填充单元格区域()
    Dim i As Long, j As Long
    Dim col As Long, row As Long
    Dim arr() As Long
    row = Application.InputBox(prompt:="输入行数: ", Type:=2)
    col = Application.InputBox(prompt:="输入列数: ", Type:=2)
    ReDim arr(row, col)
    For i = 1 To row
        For j = 1 To col
```

```
            arr(i, j) = (i - 1) * col + j
        Next
    Next
    Set Rng = Sheets(1).Range(Cells(1, 1), Cells(row, col))
    Rng.Value = arr
End Sub
```

为了运行上面的代码，在 Excel 工作表中增加一个【填充数据】按钮，单击该按钮，弹出如图 6-13 所示的对话框，分别输入数组的行和列。





图 6-13　输入行和列

运行程序 VBA 代码生成一个二维数组，最后填充到工作表中，如图 6-14 所示。

	A	B	C	D	E	F	G	H
1	1	2	3	4	5			
2	6	7	8	9	10			
3	11	12	13	14	15			
4	16	17	18	19	20			
5	21	22	23	24	25			
6	26	27	28	29	30			
7	31	32	33	34	35			
8	36	37	38	39	40			
9	41	42	43	44	45			
10	46	47	48	49	50			
11								

填充数据

图 6-14　填充数据

第 7 章　使 用 过 程

过程是一个 VBA 语句块，包含在声明语句（Function、Sub、Get、Set）和匹配的 End 声明中。

7.1　过程的相关概念

VBA 中的所有可执行语句都必须位于某个过程内。可以将整个应用程序编写为单个大的过程，但如果将它分解为多个较小的过程，代码就更容易阅读。

7.1.1　分解大过程

“结构化编程”是一种强调程序模块化和应用程序内的分层结构的方法。在 VBA 中，实现结构化编程的最直接方法是合理地使用过程将应用程序分解为离散的逻辑单元。调试各个单独的单元比调试整个程序更容易。还可以在其他程序中使用为某个程序开发的过程，而通常只需少量修改甚至不需修改。

对于复杂的程序，可将其分解为多个小过程，以方便程序的调试。将大过程分解为独立小过程的步骤如下：

（1）标识代码中一个或多个独立的部分。

（2）对于每个独立的部分，将源代码移出大过程，并用 Sub 和 End Sub 语句将其括起来。

（3）在大过程中已移除代码部分的地方，添加一个调用 Sub 过程的语句。

提示：如果新过程需要将值返回大过程，则可以定义 Function 过程。

7.1.2　过程的类型

VBA 中的过程可分为以下 3 类。

- VBA 子过程：用于执行代码后不返回值的情况，它们以关键字 Sub 开头，并以关键字 End Sub 结束。在 Excel 中录制的宏就是这种过程。
- Function 函数过程：用于执行代码后返回计算结果的情况，它们以关键字 Function 开头，以关键字 End Function 结束。使用 Function 函数过程可创建 Excel 的扩展函数。

- ❑ Property 过程：用于自定义对象。使用属性过程可设置和获取对象属性的值，或者设置对另外一个对象的引用。该类过程将在本书第 27 章进行详细介绍。

最常用的是 Function 函数过程和 Sub 子过程，它们的区别如下：

- ❑ Sub 过程不能返回值，而 Function 函数可以返回一个值，因此可以像 Excel 内部函数一样在表达式中使用 Function 函数。
- ❑ Sub 过程可作为 Excel 中的宏来调用，而 Function 函数不会出现在“选择宏”对话框中，如果要在工作表中调用 Function 函数，则可以像使用 Excel 内部函数一样使用该函数。
- ❑ 在 VBA 中，Sub 过程可作为独立的基本语句调用，而 Function 函数通常作为表达式的一部分。

注意：Function 函数也可像 Sub 一样作为独立的语句调用，只是没办法接收函数的返回值，也就失去了函数的意义了。

7.2 定义 Sub 过程

在使用模块之前，需要先对其进行定义。在 Excel 中录制宏时，录制的宏代码将自动创建一个 Sub 过程。除此之外，在 VBE 开发环境中还可以使用以下两种方式定义过程：

- ❑ 使用窗体创建过程的结构，再在过程中编写相应的代码。
- ❑ 在模块中直接输入代码来定义过程。

7.2.1 使用对话框定义子过程

VBA 开发环境中提供了一个【添加过程】对话框，通过该对话框可方便地向当前模块中添加过程。具体操作步骤如下：

（1）在 Excel 中按快捷键 Alt+F11 进入 VBE。

（2）单击主菜单【插入】|【过程】命令，将打开如图 7-1 所示的【添加过程】对话框。

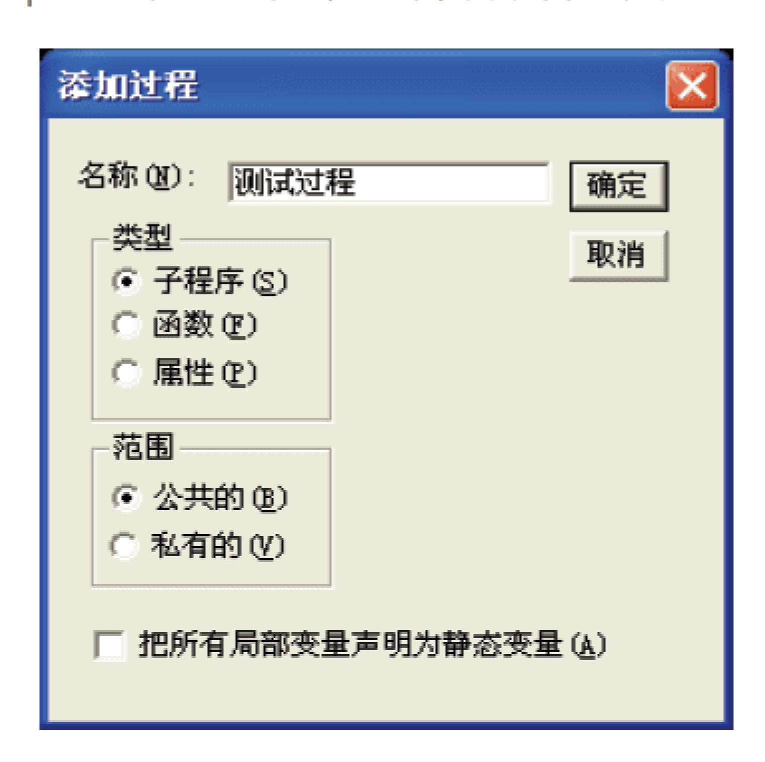


图 7-1 【添加过程】对话框

提示：使用图 7-1 所示对话框可以插入一个新的 Sub 过程，Function 过程或属性过程。还可以设置 Public 或 Private 的有效范围，并使过程中所有的区域变量成为过程变量。

（3）在【名称】文本框中输入过程的名称、在【类型】选项中选择【子程序】单选按钮。

（4）在【范围】选项中选中【公共的】单选按钮，设置过程为全局的（即在工程的各模块中都可以调用该过程），将在过程前面添加 Public 关键字。

还可根据需要选中下方的【把所有局部变量声明为静态变量】复选框，如果选中该复选框，将在过程名前面添加 Static 关键字。

（5）设置好以上参数后，单击【确定】按钮，VBA 将自动生成过程的结构代码，如图 7-2 所示。

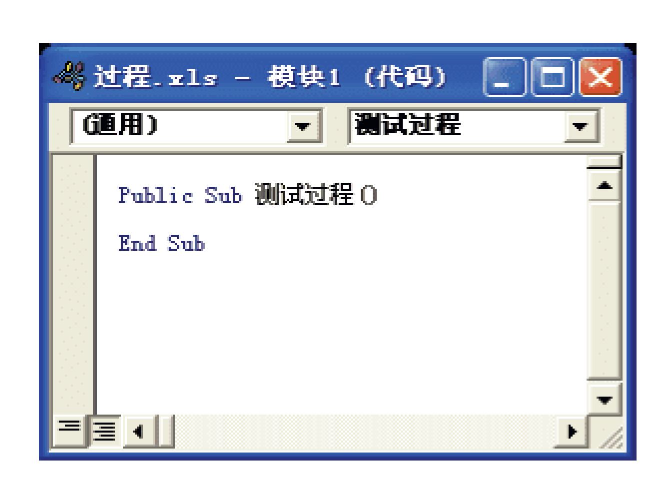


图 7-2　过程结构代码

7.2.2　使用代码创建 Sub 过程

大多数开发人员更习惯于通过手工输入的方式来创建 Sub 过程。这样更利于了解创建 Sub 过程中各元素的作用。Sub 过程的结构如下：

```
[Private | Public | Friend] [Static] Sub 过程名 [(参数列表)]
    [语句序列 1]
    [Exit Sub]
    [语句序列 2]

End Sub
```

过程由 Sub 和 End Sub 及之间的 VBA 代码来构成。其中在 Sub 前面可加上限制过程作用域的关键字，主要有以下几个。

- Private：表示只有在包含其声明模块中的其他过程可以访问该 Sub 过程。
- Public：表示所有模块的所有其他过程都可访问这个 Sub 过程。如果在 Sub 前面省略关键字，则表示其为 Public。
- Friend：只能在类模块中使用。表示该 Sub 过程在整个工程中都是可见的，但对对象实例的控制者是不可见的。
- Static：表示在调用之间保留 Sub 过程的局部变量的值。Static 属性对在 Sub 之外声明的变量不会产生影响，即使过程中也使用了这些变量。

注意：每一个过程都必须有一个名称，通过过程名称来调用该过程。过程名称的命名应符合标识符的命名规则。

End Sub 语句标志着 Sub 过程的结束。为了能正确运行，每个 Sub 过程必须有一个 End Sub 语句，当程序执行到该语句时就结束该过程的运行。另外，在过程中可以使用一个或多个 Exit Sub 语句直接退出过程的执行。

注意：一个过程中不能包含其他过程，因此其起始和结束语句必须位于其他任何过程之外。

了解 Sub 过程结构及各语句的作用后，就可以在代码窗口中创建 Sub 过程了。下面演示具体的操作步骤：

（1）进入 VBE 操作环境。

（2）双击【工程资源管理】窗口中的【模块 1】命令，打开代码窗口。

（3）在代码窗口中输入“Sub 手工创建 Sub 过程”，并按 Enter 键。

（4）系统自动在过程名后面添加一对括号，并自动生成 End Sub 语句。

（5）在过程结构中输入以下代码：

```
Sub 手工创建 Sub 过程()
    MsgBox "这是手工输入代码创建的 Sub 过程！"
End Sub
```

7.3 定义 Function 函数过程

Function 函数过程的创建方法与 Sub 过程的方法类似。在使用 Function 函数时，一般需要使用一个变量来接收返回值。

7.3.1 使用对话框定义函数过程

通过对话框创建函数过程的方法与创建 Sub 子过程相似，具体操作步骤如下：

（1）在 Excel 中按快捷键 Alt+F11 进入 VBE。

（2）单击主菜单【插入】|【过程】命令，将打开如图 7-3 所示的【添加过程】对话框。

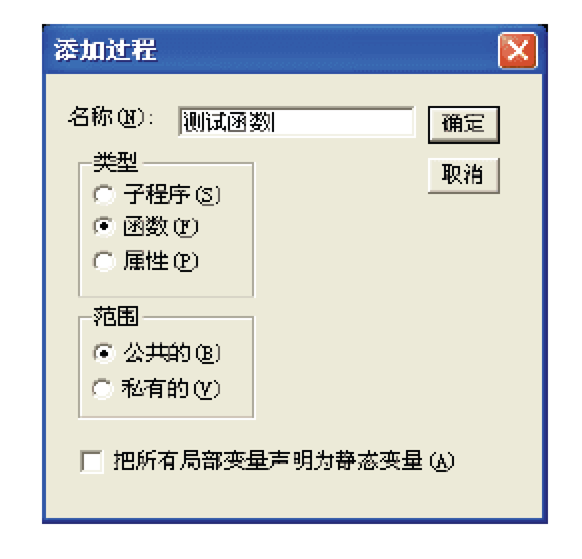


图 7-3 【添加过程】对话框

（3）在【名称】文本框中输入函数过程的名称、在【类型】选项中选择【函数】单选按钮。

（4）在【范围】选项中选择【公共的】单选按钮，设置过程为全局的（即在工程的各模块中都可以调用该函数过程），将在函数过程前面添加 Public 关键字。

还可根据需要选中下方的【把所有局部变量声明为静态变量】复选框，如果选中该复选框，将在函数过程名前面添加 Static 关键字。

（5）设置好以上参数后，单击【确定】按钮，VBA 将自动生成函数过程的结构代码，如图 7-4 所示。

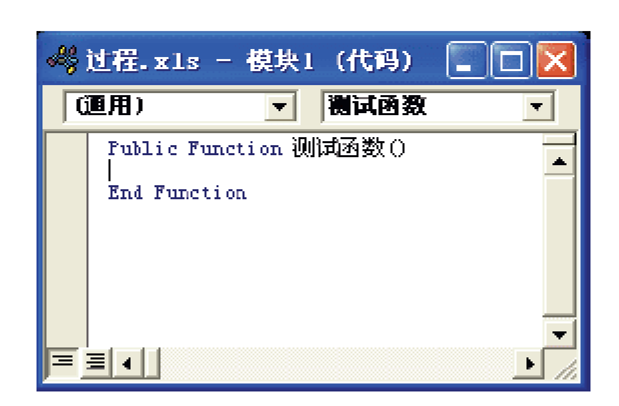


图 7-4　函数过程结构代码

7.3.2　使用代码创建 Function 过程

要手工输入代码创建 Function 函数，首先要了解 Function 结构。其语法格式如下：

```
[Public | Private | Friend] [Static] Function 函数名 [(参数列表)] [As 返回类型]
     语句序列 1
     函数名 = 表达式 1
     Exit Function
     语句序列 2
     函数名 = 表达式 2

End Function
```

可以看出，Function 函数的结构与 Sub 过程的结构很相似，下面介绍二者的不同之处：

❑ 在声明函数名的第一行使用“As 返回类型”定义函数的返回值类型。

```
Function 函数名 [(参数列表)] [As 返回类型]
```

❑ 在函数体内，通过给函数名赋值来返回计算结果。

```
函数名 = 表达式 1
```

如果在函数体内没有上面的语句，则该函数返回一个默认值：数值函数返回 0，字符串函数返回空字符串。

例如，使用以下代码定义生成指定范围内的随机整数的函数：

```
Function fTest1(a As Integer, b As Integer) As Integer
    Dim t As Integer
    Randomize
```

```
    If a > b Then
        t = a
        a = b
        b = t
    End If
    fTest1 = Int(Rnd * (b - a)) + a
End Function
```

以上函数将返回 a～b 之间的一个随机整数。

7.4　过程的调用

创建过程的目的就是将一个应用程序划分为若干个小的模块，每个小模块完成一个具体的功能，最后通过组合这些过程来完成一个大的任务。

对于 Sub 和 Function 过程，其调用方式有相同之处，也有不同之处，本节分别介绍这两种过程的调用方式。

7.4.1　调用 Sub 过程

在程序中调用 Sub 过程有两种方式：一是把过程名字放在一个 Call 语句中；另一种是把过程名作为一个语句来使用。

1．用Call语句调用Sub过程

用 Call 语句可将程序执行控制权转移到一个 Sub 过程或 Function 过程中，在过程中遇到 End Sub 或 Exit Sub 语句后，再将控制权返回到调用程序的下一行。

Call 语句的语法格式很简单：

```
Call 过程名(过程参数列表)
```

如果使用 Call 语句来调用一个需要参数的过程，则“参数列表”必须要加上括号，如果过程没有参数，则可省略过程名后的括号。

例如以下代码：

```
Call TestSub
```

将调用过程“TestSub”。

2．将过程作为一个语句

在调用过程时，如果省略 Call 关键字，也可调用过程。与使用 Call 关键字不同的是，如果过程有参数，这种调用方式必须要省略“参数列表”外面的括号。例如：

```
Call Test(a, b)
```

可改为以下形式：

```
Test  a, b
```

7.4.2　调用 Function 过程

有两种方法可以调用 Function 函数：一种是在工作表的公式中调用它，另一种是从 VBA 的另外一个过程里调用它。

1．在工作表中调用函数

自定义 Function 函数和系统内置函数一样，可以对 Excel 工作表的公式引用。如果不知道 Function 函数的名称或参数，可以使用【插入函数】对话框帮助用户向工作表中输入这些函数。

例如：在 VBA 中已经编写了一个 Function 函数 fTest，在工作表中引用该函数的过程如下：

（1）返回到 Excel 窗口，单击选择一个单元格。

（2）在【公式】选项卡的【函数库】组中，单击【插入函数】按钮，将打开如图 7-5 所示的【插入函数】对话框。在类别下拉列表中选择“用户定义”，下方的函数列表将显示定义的函数。

（3）然后选择一个自定义函数 fTest，单击【确定】按钮，将打开如图 7-6 所示的【函数参数】对话框，在该对话框中可输入函数所需要的参数。

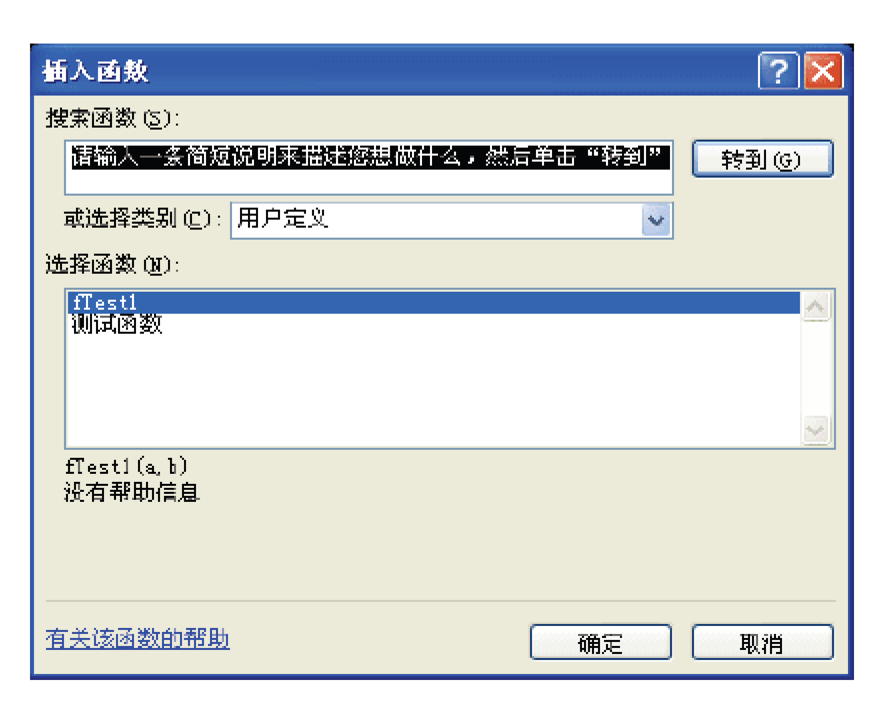



图 7-5　【插入函数】对话框

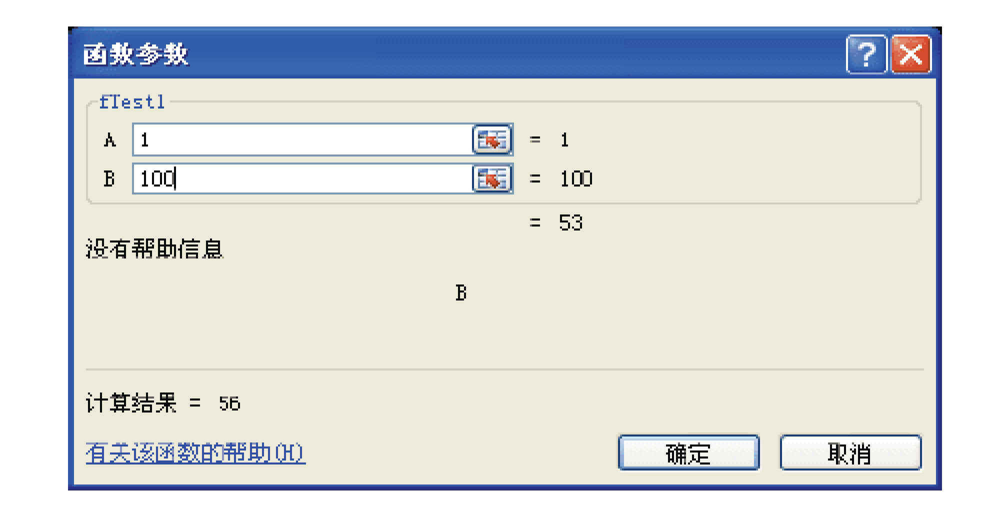


图 7-6　【函数参数】对话框

（4）输入完参数后，单击【确定】按钮，完成公式的输入。

2．在VBA代码中调用函数

在 VBE 开发环境中，不能像 Sub 过程一样按 F5 键来运行 Function 函数。必须从另一个过程里调用 Function 函数。

Function 函数的调用比较简单，可以像使用 VBA 内部函数一样来调用 Function 函数。它与内部函数没什么区别，只不过内部函数 VBA 由系统提供，而 Function 函数是由用户自己定义的。

例如，下面的代码调用 7.3.2 节定义的 fTest1 函数，并将计算结果保存到变量 R 中。

```
Sub 生成随机数()
    Dim R As Integer, I As Integer, u As Integer
    I = Val(InputBox("请输入随机数的下限：", "设置下限", 1))
    u = Val(InputBox("请输入随机数的上限：", "设置上限", 100))

    R = fTest1(I, u)
    MsgBox "生成的随机数为：" & R
End Sub
```

还可将 Function 函数作为表达式的一部分，使用其返回值参加表达式的运算。例如：

```
    r = r + fTest1(1, 100)
```

在代码中也可像调用 Sub 过程一样，直接输入函数的名称，后面是参数（参数不能用括号引起来），如以下代码：

```
    fTest1 1, 100
```

这种方式下，没有变量接收函数的返回值，所以得不到运行结果。虽然语法上没有错误，但是没有什么实际意义。

7.5 过程的参数传递

为了使过程更具有通用性，很多过程都需要设置参数，传递不同的参数给过程，可使执行的结果不同。对于有参数的过程，在调用时必须将实际参数传递给过程，完成形参与实参的结合，过程再使用实参执行代码。

7.5.1 形参与实参的结合

形参是形式参数的简称，是在 Sub 过程的定义中出现的变量名，因其没有具体的值，只是形式上的参数，所以称为形参。

实参是实际参数的简称，是在调用 Sub 过程时传递给 Sub 过程的值。在 VBA 中实参可以是常量、变量、数组或对象类的数据。

在 VBA 中，形参与实参的结合有两种方式：

1．按位置结合

大多数程序语言调用子过程时都是使用按位置结合形参与实参。在这种方式下，调用 Sub 过程时使用的实际参数次序必须与定义 Sub 过程时设置的参数次序相对应。

例如：使用以下代码定义 Sub 子过程。

```
Function fTest1(a As Integer, b As Integer) As Integer
    ……
End Sub
```

子过程中定义了两个参数，可使用以下语句调用该子过程：

```
Call fTest1(1, 100)
```

此时，Test 子过程的形参与实参的结合过程为：将常量 1 赋值给变量 a，常量 100 赋值给变量 b。

2．按命名参数方式结合

形参与实参的另一种结合方式是按形参名称来进行，即在调用过程时，输入形参的名称，将形参名称与实参用“:=”符号连接起来。与按位置结合方式不同，使用这种方式时，调用过程的参数位置可随意设置。

例如：使用命名结合的方式调用上面定义的“fTest1”过程：

```
Call fTest1(arg1:=1, arg2:=100, arg2:=2)
```

或

```
Call fTest1(arg2:=100, arg1:=1)
```

都可得到同样的效果。

按命名参数方式结合形参和实参，在输入代码时要增加一些工作量，但其好处也显而易见，通过这种方式可以改变过程调用的可读性，减少程序出错。

3．按位置和名称混合结合参数

在单个过程调用中，可以同时通过位置和名称提供参数，如下例所示：

```
Call fTest1(1, arg2:=100,)
```

在上面的代码中，由于 arg2 是通过名称传递的，所以没有必要用额外的逗号来保留省略的 age 参数的位置。

如果同时通过位置和名称提供参数，那么定位参数都必须放在前面。通过名称提供了一个参数后，其余的参数必须都通过名称提供。

7.5.2　按传值方式传递参数

在 VBA 中，实参可通过两种方式将数据传递给形参，即传地址和传值。传值就是将实参的值作为一个副本，赋值给形参（相当于执行一次赋值操作），而不是传送实参的地址给形参。定义过程时，在形参的前面添加 ByVal 关键字，则该参数就按传值方式传递；否则用传地址方式传递。

使用传值方式传递参数时，传递的只是变量的副本，如果过程改变了形参的值，那么所做的改变也只在过程内部起作用，不会影响到调用程序中变量的值。

例如，定义以下过程代码：

```
Sub 传值测试(ByVal a As Integer)
    a = a + 1
    Debug.Print "子过程中的变量 A=" & a
End Sub
```

过程中用 ByVal 关键字将参数定义为传值方式。

```
Sub 调用传值测试()
    Dim b As Integer
    b = 3
    Debug.Print "主程序中变量 B=" & b
    传值测试 b
    Debug.Print "主程序中变量 B=" & b
End Sub
```

执行以上过程后【立即窗口】列表框中显示的内容如图 7-7 所示。

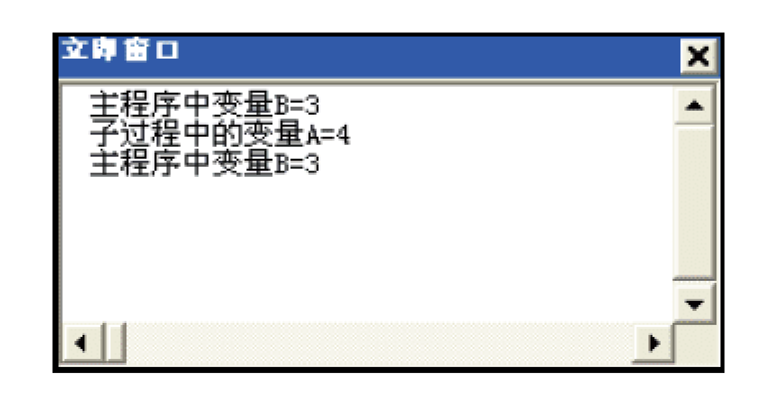


图 7-7 【立即窗口】列表框

从图 7-7 的运行结果可以看出，主程序中的变量 b 和子过程中的变量 a 之间是被隔离的，即在子过程中改变了变量 a 的值，并不会影响主程序中的变量 b 的值。

在程序中使用传地址比传值的效率高，但是在传地址方式中，形参不是一个真正的局部变量，有可能对程序产生不必要的影响。如果没有特殊要求，在过程中应尽量使用这种方式传递参数。

7.5.3 按传地址方式传递参数

传地址是 VBA 默认的方式，在定义过程时，如果在形参前面有关键字 ByRef，则该参数通过传地址的方式传递。

传地址是指将实参变量的地址传递给形参，这时形参和实参都代表同一个内存区域。所以，在过程中对形参的值进行了改变，返回到调用程序后，使用实参变量名也可访问到改变后的值。

例如：编写以下 VBA 代码，使用形参传地址方式进行传递。

```
Sub 传地址测试(ByRef a As Integer)
    a = a + 1
    Debug.Print "子过程中的变量 A=" & a
End Sub
```

使用以下过程调用“传地址测试”子过程。

```
Sub 调用传地址过程()
    Dim b As Integer
    b = 3
    Debug.Print "主程序中变量 B=" & b
    传地址测试 b
```

```
    Debug.Print "主程序中变量 B=" & b
End Sub
```

程序首先给变量 b 赋初值，并输出变量 b 的值（为 3），接着调用“传地址测试”子过程，使用传地址方式传递变量，使子过程的形参变量 a 的传地址与调用过程中的实参变量 b 的地址相同。在子过程中将变量 a 增加 1，并输入出变量 a 的值（为 4），返回调用过程后，因为变量 b 和变量 a 是指向同一内存单元的，所以变量 b 的值也为 4。

运行以上程序后，【立即窗口】列表框中的显示结果如图 7-8 所示。

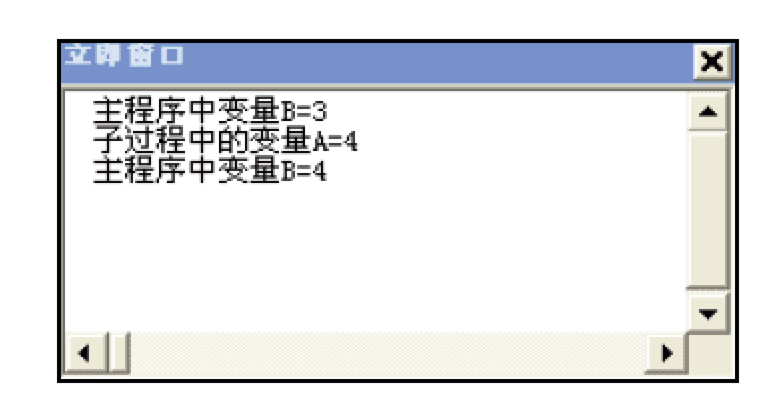


图 7-8　程序运行结果

前面说过，Sub 过程不能返回运算结果。如果需要 Sub 过程返回值时，通过使用 ByRef 方式来定义形参就可以将子过程的运算数据返回到调用程序中。

注意：当形参定义为 ByRef 形式时，只有当实参为一个变量时，才能按传地址方式传递参数，如果实参为一个表达式或常量，则不能按传地址方式传递参数。

7.5.4　传递数组参数

数组作为在内存中的一片连续区域，也可作为一个参数传递给 Sub 子过程进行处理。数组一般是通过传地址方式进行传递的。

例如：以下代码为求数组中最大数的过程。

```
Sub 求最大数(a() As Integer)
    Dim i As Integer, max As Integer
    max = a(LBound(a))
    For i = LBound(a) To UBound(a)
        If a(i) > max Then max = a(i)
    Next
    Debug.Print "最大数:" & max
End Sub
```

在该过程中，将形参定义为一个数组。使用数组作为形参时，必须输入数组名并加上一对空括号。对传递到过程中的数组，应使用 LBound 函数和 UBound 函数获取其下标的下界和上界，然后程序中才能遍历数组，或对数组进行其他相关的操作。

在本例中，将最大值保存在另一个变量 max 中，所以使用 max 和每个数组元素进行比较，如果数组元素比 max 大，则将其值赋值给 max，确保 max 中始终保存着最大的数。

编写调用“求最大数”的代码如下：

```
Sub 测试求最大数()
    Dim MyArray(5) As Integer, i As Integer
    For i = 0 To 5
        MyArray(i) = i * 2
    Next
    求最大数 MyArray()
End Sub
```

在程序中，调用“求最大数”子过程，并将数组 MyArray 作为参数传递到该子过程中进行处理。

7.6 可选参数和可变参数

在创建 VBA 的过程时，除了可使用前面介绍的方法设置参数按地址或按值传递之外，还可以根据需要为过程设置可选参数和可变参数。

7.6.1 可选参数

通常，一个 VBA 过程中的形参数量是固定的，调用时提供的实参数量也是固定的。在有些过程中，可能有必须收集的信息和可选信息，如收集顾客的信息时必须提供姓名、性别，对于身份证号码则是可选的（可提供，也可不提供）。

可以指定过程参数是可选的，并且在调用过程时不必为其提供变量。“可选参数”在过程定义中由关键字 Optional 指示。适用以下规则：

- ❑ 过程定义中的每个可选参数都必须指定默认值。
- ❑ 可选参数的默认值必须是一个常数表达式。
- ❑ 过程定义中跟在可选参数后的每个参数也都必须是可选的。

在过程内部通过使用 IsMissing 函数来测试调用程序是否传递了该可选参数。

例如：以下代码定义可选参数：

```
Sub 可选参数(strName As String, strSex As String, Optional ID)
    With Worksheets("sheet2")
        .Range("A2") = strName
        .Range("B2") = strSex
        If Not IsMissing(ID) Then
            .Range("C2") = ID
        End If
    End With
End Sub
```

该过程有三个参数，前两个参数为必须使用的，最后一个参数使用了 Optional 关键字，表示该参数是可选参数。调用该过程时，可以提供 2 个参数，也可以提供 3 个参数。

```
可选参数 "王小凤", "女"
```

或：

```
可选参数 "王小凤", "女", "410214198501012345"
```

注意：过程中可定义多个可选参数，但可选参数必须放在参数表的最后，而且必须是 Variant 类型。

7.6.2　可变参数

无论是固定参数还是可选参数，在定义过程时都已经指定了参数的个数。在 VBA 中，还可以定义可变参数，即参数的个数在定义时是未知的。

在定义过程的参数表时，在最后一个参数前面加上 ParamArray 关键字，过程将接受任意个数的参数。

注意：ParamArray 关键字不能与 ByVal，ByRef，或 Optional 一起使用。且使用 ParamArray 定义的可变参数必须为 Variant 数组。

例如，使用可变参数编写求和函数 SUM。

```
Sub MySum(intTotal As Integer, ParamArray intNum())
    Dim i As Integer, j As Integer
    For i = LBound(intNum) To UBound(intNum)
        intTotal = intTotal + intNum(i)
    Next
End Sub
```

程序中定义了两个形参，第一个用于返回计算的结果，第二个参数使用 ParamArray 关键字将其定义为可变参数。

可变参数为一个数组，在程序中使用 LBound 函数和 UBound 函数获得数组下标的上下界，然后对其进行累加，并将累加的结果保存在第一个参数中，用于返回给调用程序。

调用以上子过程的代码如下：

```
Sub 调用可变参数()
    Dim i As Integer
    MySum i, 1, 2, 3, 4, 5, 6, 7, 8
    Debug.Print i
End Sub
```

在程序中定义了一个变量 i，用来获得子过程运算的结果。

参数传递时，将参数 i 用传地址方式传递给 MySum 子过程的形参 intTotal，将后面的“1，2，3，4，5，6，7，8”作为一个数组传递给 intNum 形参。

注意：ParamArray 只能用于参数列表的最后一个参数，指明最后这个参数是一个 Variant 元素的 Optional 数组。ParamArray 关键字不能与 ByVal，ByRef 或 Optional 一起使用。

7.7 递归过程

“递归”过程是指调用自身的过程。在递归调用中，一个过程执行的某一步要用到它自身的上一步（或上几步）的结果。

VBA 的 Sub 过程和 Function 函数过程都支持递归调用。递归调用分为两种类型：

- 直接递归，即在过程中调用过程本身。
- 间接递归，即间接地调用一个过程，如过程 1 调用过程 2，过程 2 又调用过程 1。

通常，这不是编写 VB 代码的最有效方法。

在执行递归操作时，VBA 把递归过程中的信息保存在堆栈中，过程执行结束时从堆栈中获取调用过程中各变量的数据。

阶乘运算是一个典型的递归操作，下面编写 VBA 代码来完成该操作。

```
Function fact (ByVal n As Integer) As Integer
    If n <= 1 Then
        Return 1
    Else
        Return fact (n - 1) * n
    End If
End Function
```

在以上代码中，假设参数 n 为 5，程序执行的过程如表 7-1 所示。

表 7-1　递归过程

递 归 次 数	操　作	返　回	结　果
0	fact(5)	1*2*3*4*5	120
1	fact(4)	1*2*3*4	24
2	fact(3)	1*2*3	6
3	fact(2)	1*2	2
4	fact(1)	1	1

使用递归过程时，应注意以下几点：

- 限制条件。在设计一个递归过程时，必须至少测试一个可以终止此递归的条件，并且还必须对在合理的递归调用次数内未满足此类条件的情况进行处理。如果没有一个在正常情况下可以满足的条件，则过程将陷入执行无限循环的高度危险之中。
- 内存使用。应用程序的局部变量所使用的空间有限。过程在每次调用自己时，都会占用更多的内存空间以保存其局部变量的附加副本。如果这个进程无限持续下去，最终会导致 StackOverflowException 错误。
- 效率。几乎在任何情况下都可以用循环替代递归。循环不会产生传递变量、初始化附加存储空间和返回值所需的开销，因此使用循环相对于使用递归调用可以大

幅提高性能。

- ❑ 相互递归。如果两个过程相互调用，可能会使性能变差，甚至产生无限循环。此类设计所产生的问题与单个递归过程所产生的问题相同，但更难检测和调试。
- ❑ 调用时使用括号。当 Function 过程以递归方式调用自己时，必须在过程名称后加上括号（即使不存在参数列表）。否则，函数名就会被视为表示函数的返回值。
- ❑ 测试。在编写递归过程时，应非常细心地进行测试，以确保它总是能满足某些限制条件。还应该确保不会因为过多的递归调用而耗尽内存。

7.8　常用过程实例

前面介绍了 Sub 过程和 Function 函数过程的相关内容。本节介绍一些常用的过程实例代码，可直接应用到 Excel 应用程序中。

7.8.1　计算个人所得税

在工资管理系统中，需要计算员工应缴纳的个人所得税。我国个人所得税税额按 5%至 45%的 9 级超额累进税率计算应缴税额，税率表如图 7-9 所示。个人所得税的计算公式为：

```
应纳个人所得税税额=应纳税所得额×适用税率-速算扣除数
```

个人所得税税率表

级数	全月应纳税所得额	税率%	速算扣除法（元）
1	不超过500元的	5	0
2	超过500元至2000元的部分	10	25
3	超过2000元至5000元的部分	15	125
4	超过5000元至20000元的部分	20	375
5	超过20000元至40000元的部分	25	1375
6	超过40000元至60000元的部分	30	3375
7	超过60000元至80000元的部分	35	6375
8	超过80000元至100000元的部分	40	10375
9	超过100000元的部分	45	15375

图 7-9　个人所得税税率表

根据图 7-9 所示的税率表编写以下函数，用来计算出个人所得税：

```
Public Function taxes(curP As Currency, Optional dep As Integer = 2000)
    Dim curT As Currency
    curP = curP - dep  '1600 为扣除数
    If curP > 0 Then
        Select Case curP
            Case Is <= 500
                curT = curP * 0.05
            Case Is <= 2000
```

```
                curT = (curP - 500) * 0.1 + 25
            Case Is <= 5000
                curT = (curP - 2000) * 0.15 + 125
            Case Is <= 20000
                curT = (curP - 5000) * 0.2 + 375
            Case Is <= 40000
                curT = (curP - 20000) * 0.25 + 1375
            Case Is < 60000
                curT = (curP - 40000) * 0.3 + 3375
            Case Is < 80000
                curT = (curP - 60000) * 0.35 + 6375
            Case Is < 100000
                curT = (curP - 80000) * 0.4 + 10375
            Case Else
                curT = (curP - 100000) * 0.45 + 15375
        End Select
        taxes = curT
    Else
        taxes = 0
    End If
End Function
```

编写好以上代码后，默认情况下，速扣数为 2000，可在单元格中输入以下公式计算个人所得税：

```
=taxes(3000)
```

如果调整了速扣数（速扣数不为 2000），则可使用以下格式调用 taxes 函数，计算出个人所得税：

```
=taxes(3000, 2500)
```

7.8.2 将数值转换为表格的列号

在 Excel 工作表中，用户可使用 A1 样式访问单元格，这种方式的列用字母或字母组合来表示。有时用户可能想知道单元格所在列的序号，可以在 Excel 中可按以下方式获得：

```
Sub 显示列号()
    With Worksheets("sheet1")
        .Activate
        .Range("AZ1").Select
    End With
    MsgBox Selection.Column
End Sub
```

但是，在 Excel 中没有内置的函数或方法获取指定列号的字母。因此，需要用户另外编写函数。下面的函数即可完成该功能。

```
Public Function NumtoCol(Numbers As Integer) As String
```

```
    Dim i1 As Integer, i2 As Integer, i3 As Integer
    Dim s1 As String, s2 As String, s3 As String
    i2 = Numbers \ 26
    i3 = i2 \ 26                                        '第 3 位
    i2 = i2 Mod 26                                      '第 2 位
    i1 = Numbers Mod 26                                 '第 1 位
    If i2 > 0 And i1 = 0 Then
        i1 = 26
        i2 = i2 - 1
    End If
    If i3 > 0 And i2 = 0 Then
        i2 = 26
        i3 = i3 - 1
    End If
    s3 = Chr(i3 + 64)
    s2 = Chr(i2 + 64)
    s1 = Chr(i1 + 64)

    If s3 = "@" Then
        If s2 = "@" Then
            NumtoCol = s1
        Else
            NumtoCol = s2 & s1
        End If
    Else
        NumtoCol = s3 & s2 & s1
    End If
End Function
```

编写以下 VBA 代码，测试上面的函数：

```
Sub 测试显示列号()
    Dim intCol As Integer
    intCol = Val(InputBox("请输入列号（1～16384）: "))
    If intCol < 1 Or intCol > 16384 Then
        MsgBox "输入的数据超过范围，请重新输入！"
        Exit Sub
    End If

    MsgBox "列号: " & intCol & ", 对应的字母为: " & NumtoCol(intCol)
End Sub
```

运行以上过程，显示如图 7-10 所示对话框，输入一个列号后，将显示如图 7-11 所示的结果。

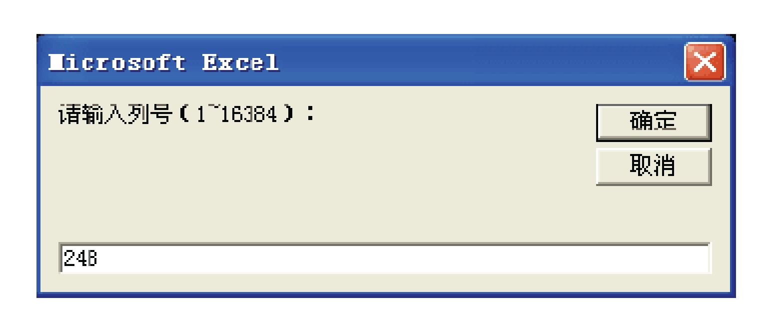


图 7-10　输入列号



图 7-11　程序运行结果

7.8.3 大写金额转换函数

在用 Excel 进行财务统计时，需要将金额由阿拉伯数字转换为中文大写形式，通常是将单元格格式自定义为“[dbnum2]&元”。但是对带有小数的数值却不能得到正确的结果，如在单元格中输入 1234.5，转换的结果却为“壹仟贰佰叁拾肆．伍元”。

可以通过编写 Function 函数生成正确的中文大写金额格式。具体代码如下：

```
Function CapsMoney(curMoney As Currency) As String      '转换中文大写金额函数
    Dim curMoney1 As Long
    Dim i1 As Long                                      '保存整数部分(元部分)
    Dim i2 As Integer                                   '保存十分位(角部分)
    Dim i3 As Integer                                   '保存百分位(分部分)
    Dim s1 As String, s2 As String, s3 As String        '保存转换后的字符串
    curMoney1 = Round(curMoney * 100)                   '将金额扩大 100 倍,并
                                                        进行四舍五入
    i1 = Int(curMoney1 / 100)                           '获取元部分
    i2 = Int(curMoney1 / 10) - i1 * 10                  '获取角部分
    i3 = curMoney1 - i1 * 100 - i2 * 10                 '获取分部分

    s1 = Application.WorksheetFunction.Text(i1, "[dbnum2]")
                                                        '将元部分转为中文大写
    s2 = Application.WorksheetFunction.Text(i2, "[dbnum2]")
                                                        '将角部分转为中文大写
    s3 = Application.WorksheetFunction.Text(i3, "[dbnum2]")
                                                        '将分部分转为中文大写
    s1 = s1 & "元"                                      '整数部分
    If i3 <> 0 And i2 <> 0 Then                         '分和角都不为 0
        s1 = s1 & s2 & "角" & s3 & "分"
        If i1 = 0 Then                                  '元部分为 0
            s1 = s2 & "角" & s3 & "分"
        End If
    End If
    If i3 = 0 And i2 <> 0 Then                          '分为 0,角不为 0
        s1 = s1 & s2 & "角整"
        If i1 = 0 Then                                  '元部分为 0
            s1 = s2 & "角整"
        End If
    End If
    If i3 <> 0 And i2 = 0 Then                          '分不为 0,角为 0
        s1 = s1 & s2 & s3 & "分"
        If i1 = 0 Then                                  '元为 0
            s1 = s3 & "分"
        End If
    End If
    If Right(s1, 1) = "元" Then s1 = s1 & "整"          '为"元"后加上一个
                                                        "整"字
    CapsMoney = s1
End Function
```

以上 Function 函数将传递进来的数值分割为 3 部分：元、角、分，将这几个部分单独转换为[dbnum2]格式，再根据出现的各种可能将这 3 部分与字符“元”、“角”、“分”进行组合。在进行组合时，应分别考虑元、角、分各自为 0 时的情况。

在 VBA 中编写好函数后，应在 Excel 工作表中进行测试。下面是测试的步骤：

（1）切换到 Excel 界面中。

（2）选中一个单元格，在【公式】选项卡的【函数库】组中，单击【插入函数】按钮，打开【插入函数】对话框，如图 7-12 所示。

（3）选中 CapsMoney 函数，单击【确定】按钮，打开【函数参数】对话框，选择单元格“A1”作为参数，如图 7-13 所示。

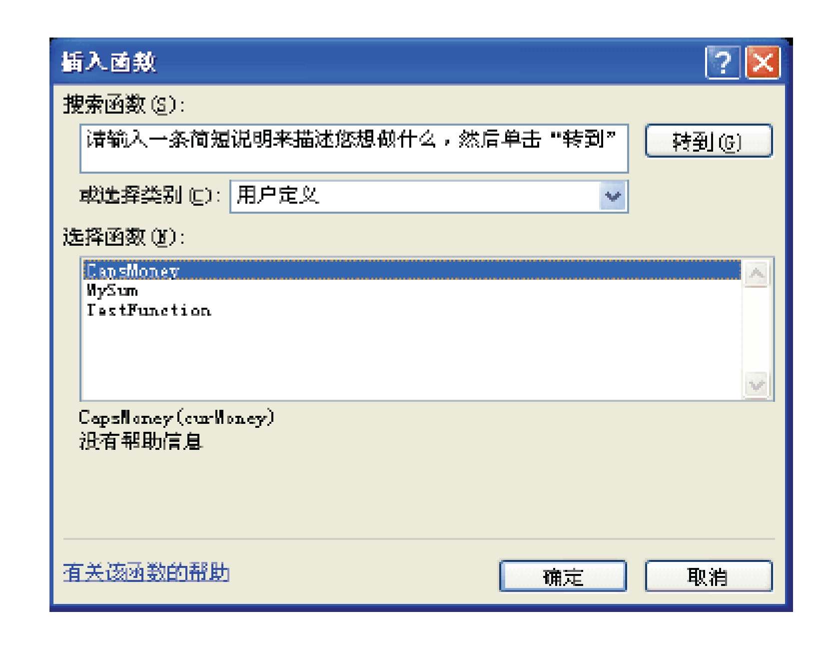
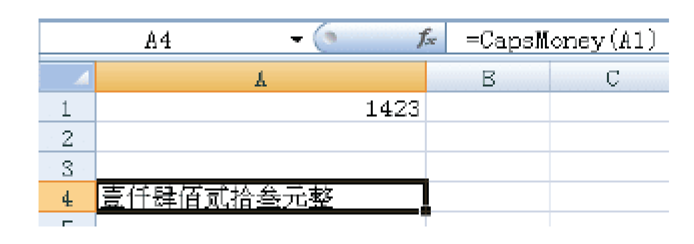
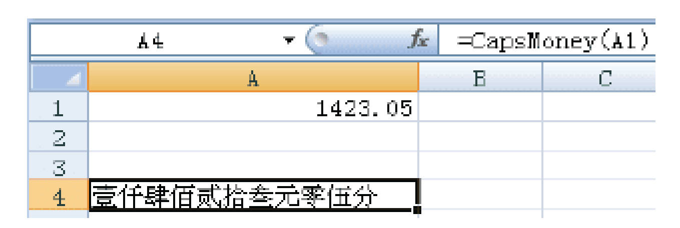


图 7-12 【插入函数】对话框

图 7-13 【函数参数】对话框

（4）在单元格“A1”中输入各种数字，检查经过函数转换后的结果是否正确，如图 7-14 所示。





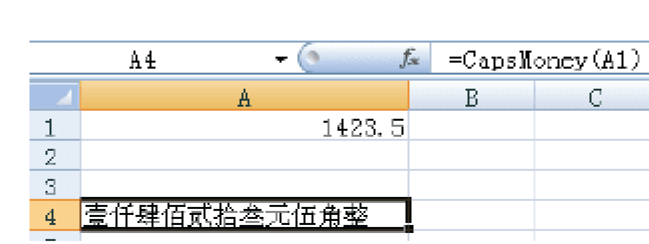


图 7-14 测试函数

第 8 章　管 理 模 块

将软件按照一定的原则划分为一个较小的、相对独立但又相关的模块，叫做模块化设计。在 VBA 中，也提供了模块化的开发方法。

在第 7 章中，介绍了过程的概念及定义方法，在一个模块中可包含多个相关的过程。本章介绍 VBA 中模块的相关知识。

8.1　模块的分类

在 VBA 中，将每一个 Excel 工作簿文件作为一个工程，统一在工程资源管理器窗口中进行管理。如图 8-1 所示，在【工程资源管理器】窗口可以看到，Excel 工程按模块管理应用程序的代码，可包含 4 类模块，每类模块中可包含多个模块。



图 8-1　工程资源管理器

1. Microsoft Excel对象

Excel 工作簿中的每个工作表都是一个单独的模块。用户可在 Excel 工作表模块中编写代码，控制工作表中的数据（通过其他模块也可控制工作表中的数据，但若要在 VBA 中捕获用户操作工作表时的事件，则必须在工作表模块中编写代码）。

有关 Excel 对象的使用内容将在本书后续章节中进行介绍。

2. 窗体

窗体模块为用户自定的对话框或界面，用来获取用户的输入，或显示输出结果。在 VBA

工程中以“.frm”为文件扩展名的文件，其中包含窗体的图形描述；其控件以及控件的属性设置；常数、变量和外部过程的窗体（模块）及声明；事件和通用过程。

本书第 4 部分将介绍创建用户窗体的方法。

3．模块

模块又称为标准模块，该类模块只包含过程、类型以及数据的声明和定义。在标准模块中，模块级别声明和定义都被默认为 Public。

在 Excel 环境中录制的宏将保存在【模块】中。

4．类模块

类模块用来保存以类或对象方式编写的代码，包括其属性和方法的定义。通过创建类模块，在 VBA 中也可以创建自己的类和对象。

本书第 27 章中将介绍创建类模块的方法。

8.2　管理标准模块

本节介绍在 VBE 中管理标准模块的方法，包括插入模块、删除模块等操作方法。

8.2.1　插入模块

在 VBE 中，用模块来组织代码。向工程中插入模块时，将自动生成名称“模块 n”（其中 n 为自动增加的一个数值）。有多种方法向 VBE 中插入模块，常见的方法如下：

1．录制宏时插入模块

在 Excel 环境中录制宏时，系统将自动在工程中插入一个 VBA 模块，并将录制的宏代码放在该模块中。

技巧：在 Excel 工作簿打开期间，录制的宏都将放在一个模块中。如果关闭该工作簿，下次再打开并录制宏时，Excel 又将插入一个新的模块。

2．使用菜单插入模块

在 VBE 环境中，单击主菜单【插入】|【模块】命令，可向工程资源管理器中插入一个模块。

3．使用快捷菜单插入模块

在 VBE 环境中，右击【工程资源管理器】窗口的某个对象，将弹出快捷菜单，如图 8-2 所示，执行【插入】|【模块】命令，可向工程资源管理器中插入一个模块。

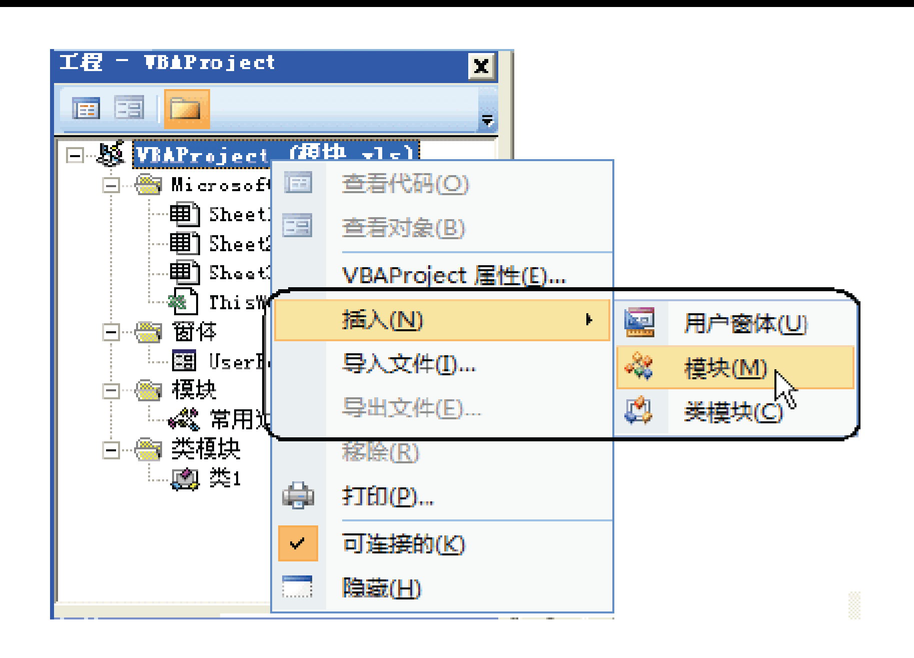


图 8-2　快捷菜单

8.2.2　删除模块

如果准备从工程中删除一个模块，可按以下步骤操作：

（1）右击【工程】资源管理器窗口中需要删除模块的名称“模块 1”，将弹出如图 8-3 所示的快捷菜单。

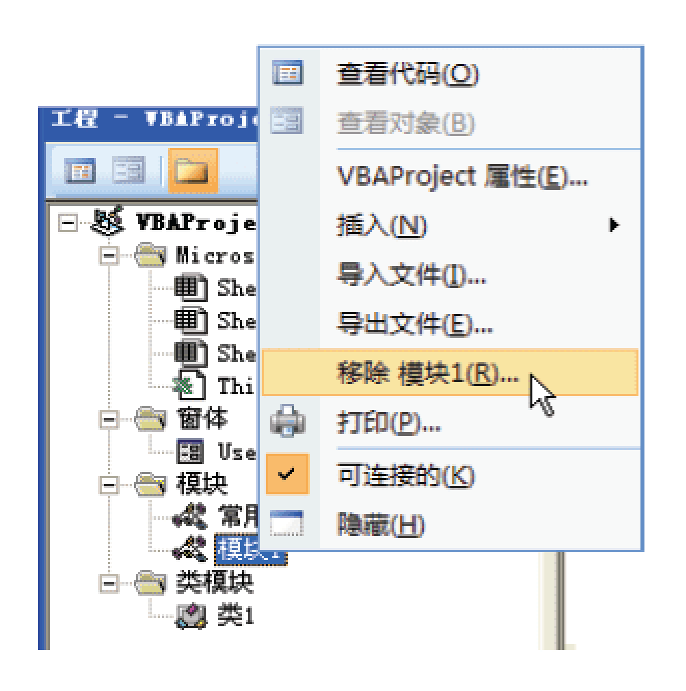


图 8-3　删除模块

（2）在快捷菜单中选择【移除　模块 1】命令即可将该模块删除。其中的“模块 1”是准备删除的模块名称。

技巧：在 VBE 环境中，选择需要删除的模块“模块 1”，再单击主菜单【文件】|【移除　模块 1】命令也可从工程资源管理器中删除“模块 1”。

8.3　模块的导入导出

正常情况下，Excel 应用程序中的模块与 Excel 工作簿保存在一起，用户只有打开工作簿后才能查看模块的内容。

对于一些具有通用性过程的模块，可将其导出保存为单独的文件，在需要的工程中再导入，可节省应用程序的开发时间。

8.3.1　导出模块

导出模块的操作步骤如下：

（1）在 Excel 2007 中打开工作簿，该工作簿包含有要导出的模块。

（2）按快捷键 Alt+F11 进入 VBE 环境。

（3）右击【工程】资源管理器窗口中需要导出的模块名称，将弹出如图 8-4 所示的快捷菜单。

（4）在快捷菜单中选择【导出文件】命令，将打开如图 8-5 所示的对话框，输入导出文件的名称“导出模块”后，单击【保存】按钮即可将选中模块的代码保存为一个扩展名为“.bas”的 Basic 代码文件。

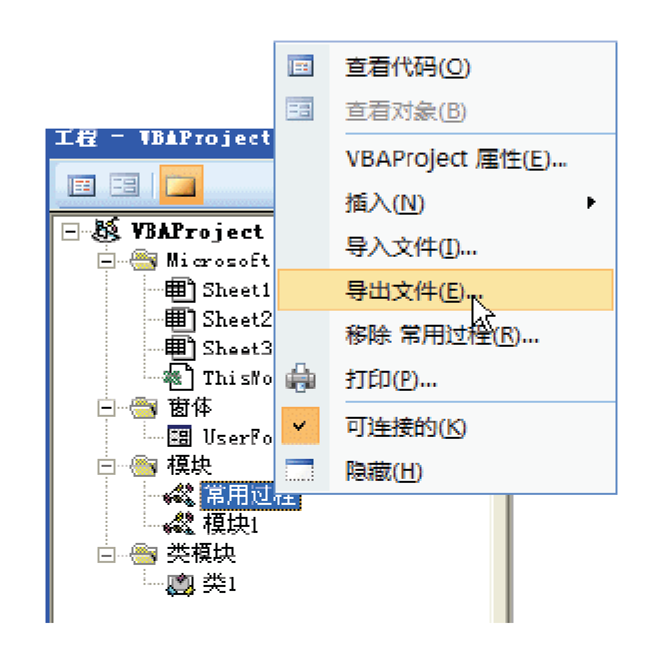


图 8-4　【导出文件】命令

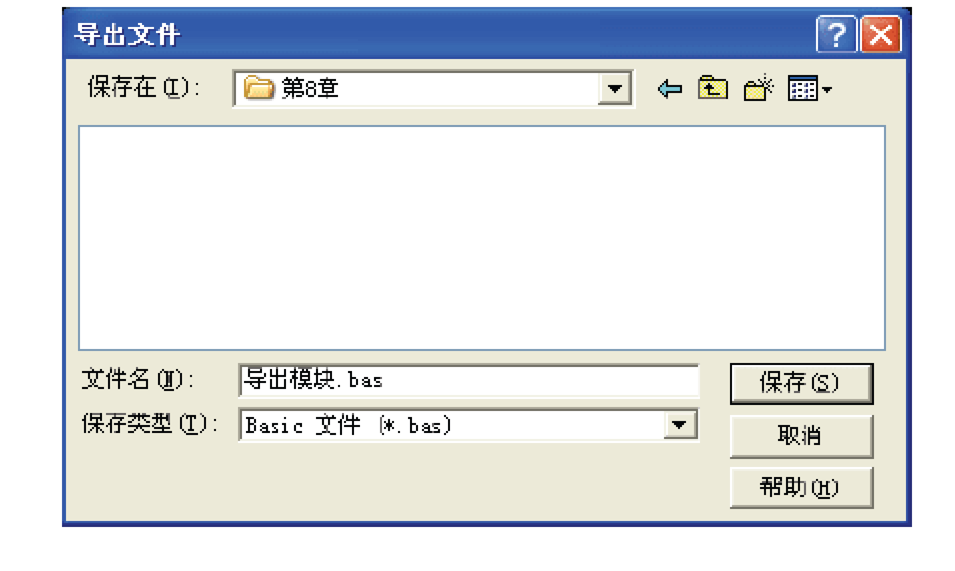


图 8-5　保存文件

（5）导出的扩展名为“.bas”的文件为文本文件格式，可用记事本打开，查看其中的代码，如图 8-6 所示。

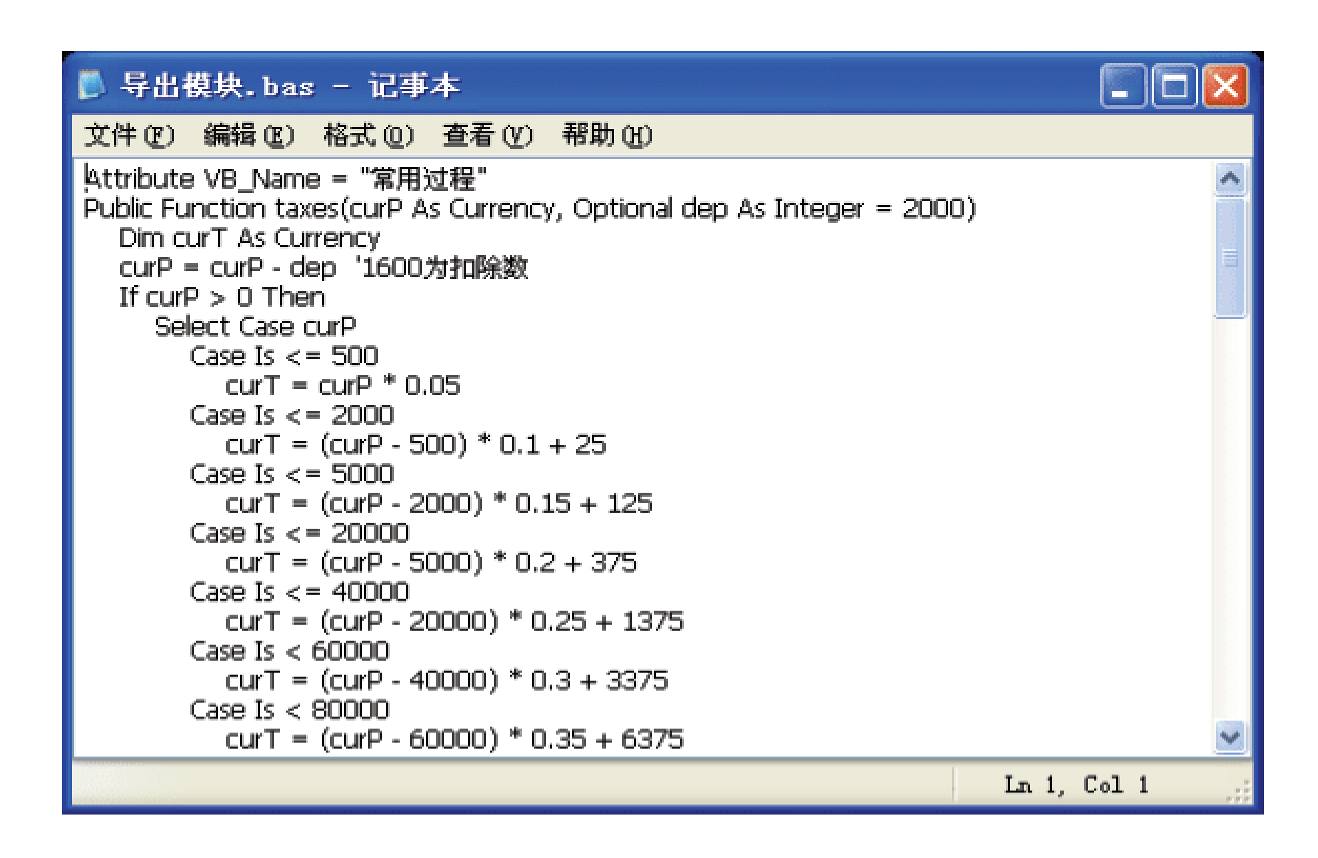


图 8-6　导出模块的代码

技巧：导出的对象不同，其扩展名也不同，对于模块其扩展名为“.bas”；对于窗体其扩展为“.frm”；对于类模块其扩展名为“.cls”。

8.3.2 导入模块

与导出操作相反，导入模块操作可将导出的模块置入当前工程中。对于导入的模块，其使用方法与在工程中编写的代码（或设计制作窗体）完全相同，因此对于一些通用模块使用导入的方法可显著提高应用程序的开发效率。将模块导入到当前工程中有两种方式：

- 将其他工程的模块导入当前工程中。
- 将其他工程导出的模块导入当前工程中。

1．导入其他工程的模块

导入其他工程模块的操作步骤如下：

（1）在 Excel 2007 中打开两个 Excel 工作簿（分别为源模块所在工作簿“模块.xls”和需要导入模块的工作簿“导入模块.xls”）。

（2）按快捷键 Alt+F11 进入 VBE 环境。

（3）在【工程】资源管理器窗口中展开“模块.xls”，找到“常用过程”模块，拖动到“导入模块.xls”列表上，如图 8-7 左图所示。松开鼠标可看到“导入模块.xls”下也新增了“常用过程”模块，如图 8-7 右图所示。

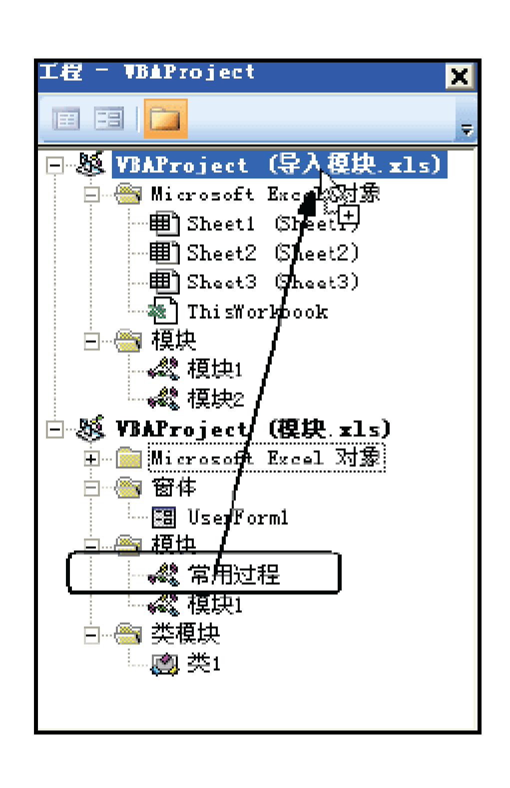




图 8-7 拖动模块

2．导入其他工程导出的模块

将其他工程导出的模块导入当前工程中，具体操作步骤如下：

（1）在 Excel 2007 中打开工作簿“导入模块.xls”。

（2）按快捷键 Alt+F11 进入 VBE 环境。

（3）右击【工程】资源管理器窗口中的工程名称“导入模块.xls”，将弹出如图 8-8 所示的快捷菜单。

（4）在快捷菜单中选择【导入文件】命令，将打开【导入文件】对话框，如图 8-9 所示。选择需要导入的文件“导出模块.bas”。

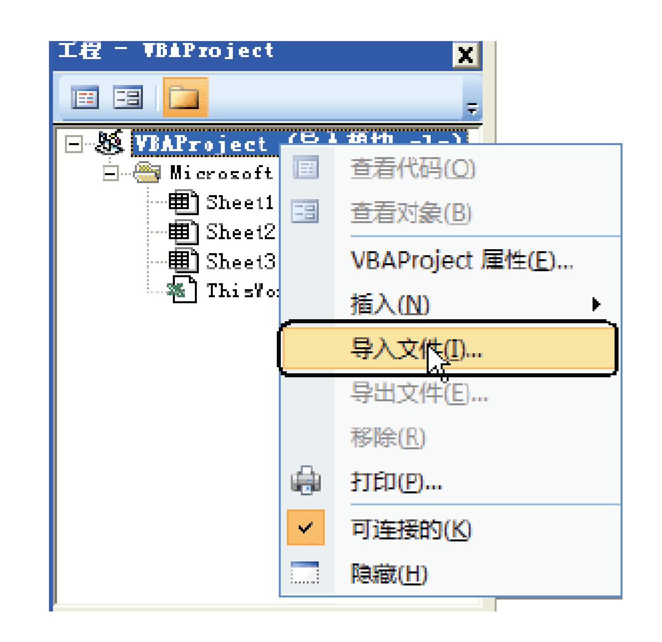


图 8-8　快捷菜单

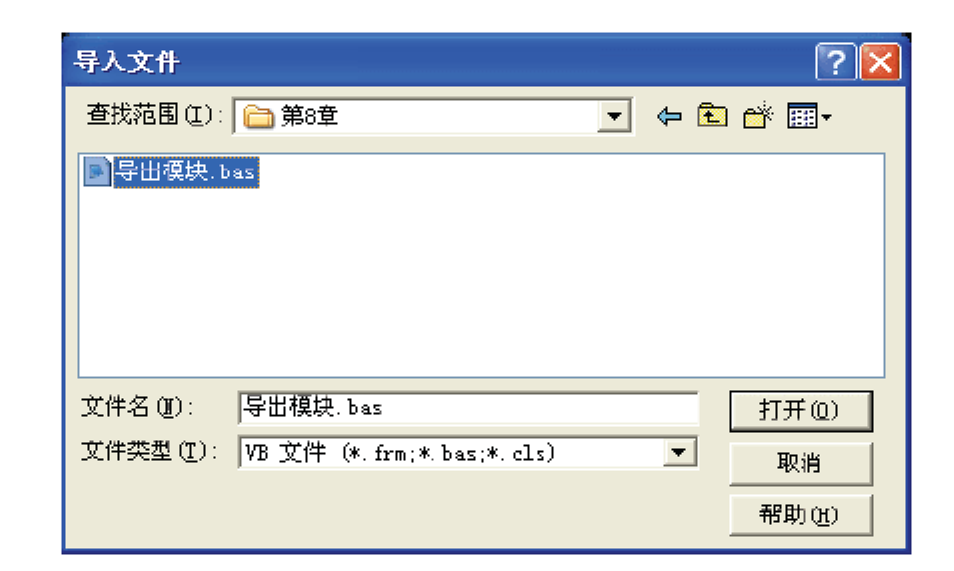


图 8-9　【导入文件】对话框

提示：在图 8-9 所示的对话框中，在【文件类型】下拉列表框中分别列出了扩展名为“.frm”和“.cls”的类型，可用来导入窗体和类模块中的代码。

（5）单击【打开】按钮即可将该文件的代码导入到当前工程中，如图 8-10 所示。

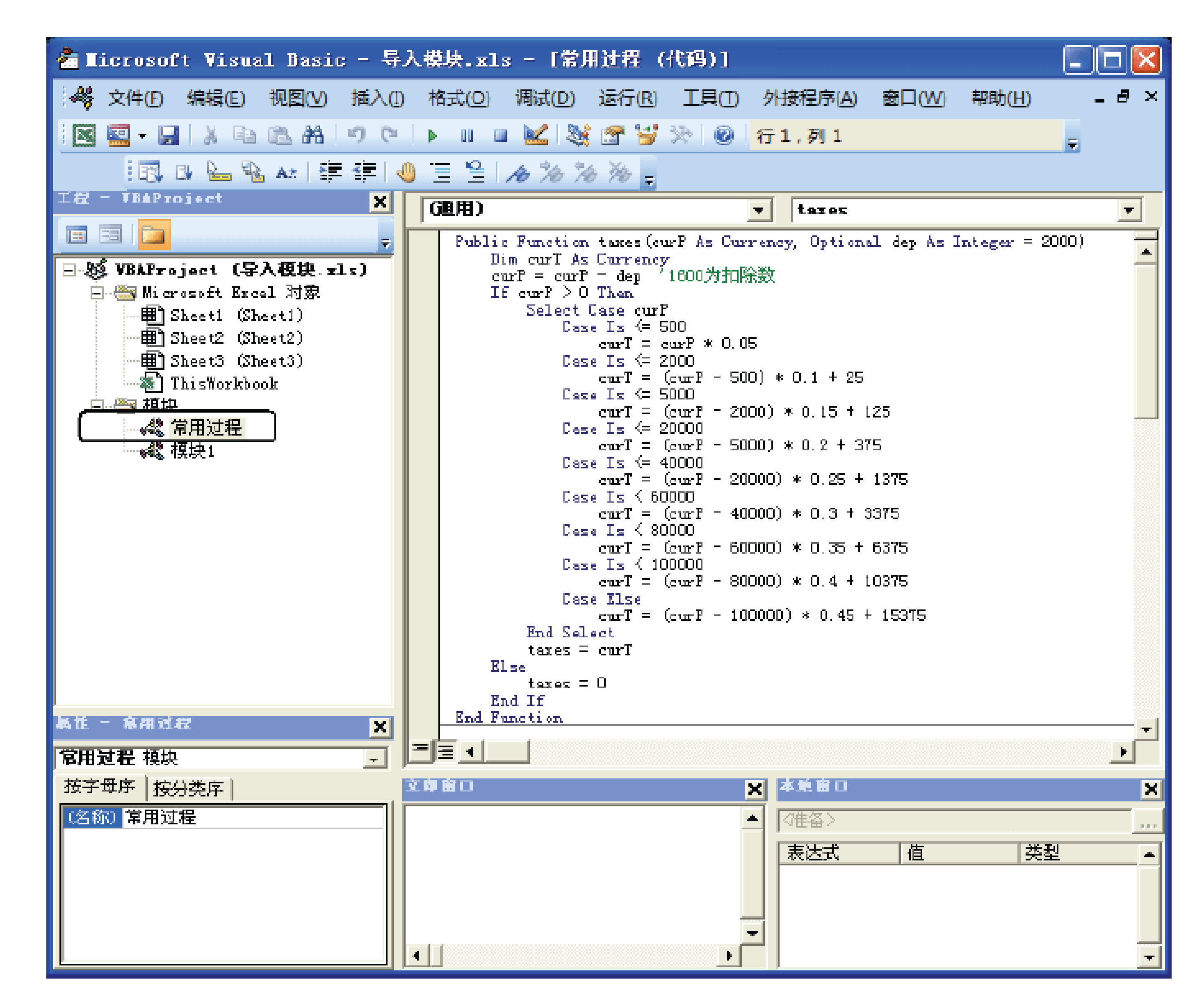


图 8-10　导入模块的代码

8.4　使用代码窗口

在 VBA 工程的各种模块（Excel 工作表对象、窗体、模块和类模块）中都可以编写 VBA 代码，这些模块都有自己的代码窗口。VBE 的代码窗口像一个专门的文字处理软件，提供了许多便于编写 VBA 代码的快捷功能。这些代码窗口的使用方法都相同，本节主要介绍代码窗口的使用技巧。

8.4.1　代码编辑工具栏

VBE 的【编辑】工具栏是专为代码窗口使用的。一般情况下，【编辑】工具栏是隐藏的。可单击主菜单【视图】|【工具栏】|【编辑】命令将其显示出来。如图 8-11 所示的【编辑】工具栏上有很多按钮，可帮助用户快速地输入、编写、编辑 VBA 代码。

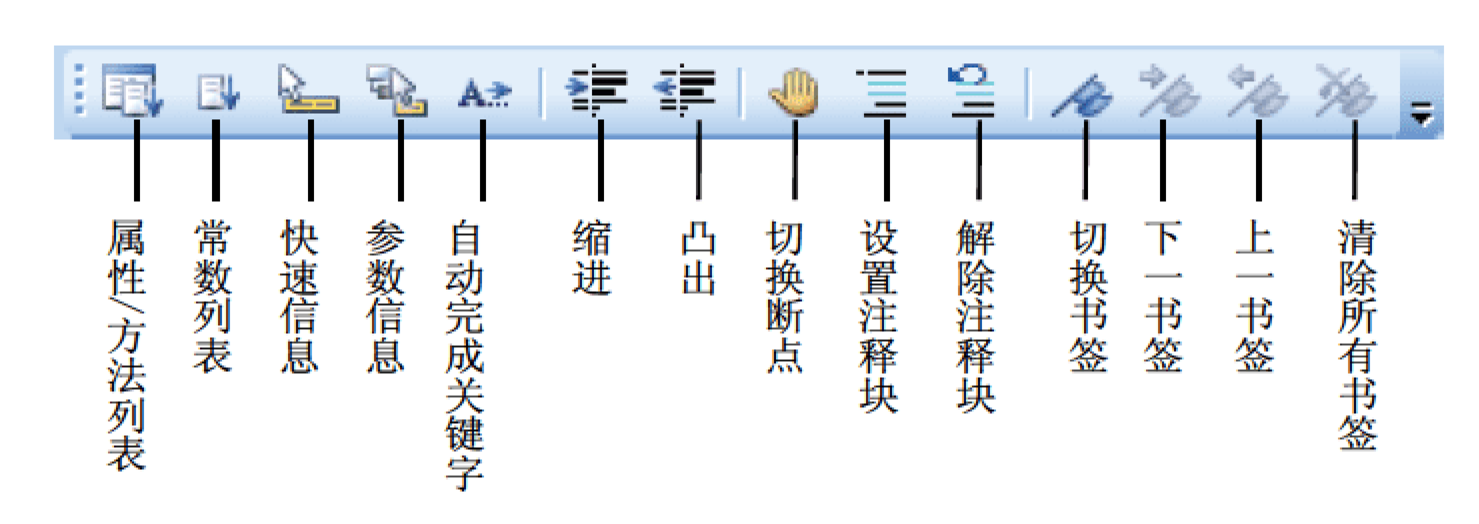


图 8-11　【编辑】工具栏

【编辑】工具栏中各按钮的功能如下所述。

- 属性/方法列表：在代码窗口中打开列表框，其中含有前面带有句点（.）的该对象可用的属性及方法。
- 常数列表：在代码窗口中打开一列表框，其中含有所键入属性的可选常数及前面带有等号（=）的常数。
- 快速信息：根据指针所在的函数、方法或过程的名称提供变量、函数、方法或过程的语法。
- 参数信息：在代码窗口中显示快捷菜单，其中包含指针所在函数的参数的有关信息。
- 自动完成关键字：接受 VB 在所键入字之后自动添加的字符。
- 缩进：将所有选择的程序行移到下一个定位点。
- 凸出：将所有选择的程序行移到前一个定位点。
- 切换断点：在当前的程序行上设置或删除一个断点。
- 设置注释块：在所选文本区块的每一行开头处添加一个注释字符。
- 解除注释块：在所选文本区块的每一行处删除注释字符。
- 切换书签：在程序窗口中使用的程序行添加或删除书签。
- 下一个书签：将焦点移到书签堆栈中的下一个书签。
- 上一个书签：将焦点移到书签堆栈中的上一个书签。
- 清除所有书签：删除所有书签。

8.4.2　属性/方法列表

使用 VBA 开发应用程序时，需要调用 Excel 中的许多对象（如单元格、图表等）。这些对象包含许多属性和方法。如果手工输入对象的方法和属性，会很容易输入错误的单词

拼写，导致程序运行出错。

在 VBE 环境中，为开发人员提供了辅助工具——属性/方法列表。开发人员在代码窗口中输入对象名称和一个句点时，将弹出一个下拉列表，该列表中列出了对象所有可用的属性和方法。

使用【属性/方法列表】的方法如下：

（1）启动 Excel 2007，或新建一个工作簿。

（2）按快捷键 Alt+F11 进入 VBE 开发环境。

（3）单击主菜单【插入】|【模块】命令向工程中插入一个模块。

（4）在【模块 1】中输入一个过程 test，在过程中输入对象 sheet1 和一个句点，将弹出如图 8-12 所示的【属性/方法】列表框。

提示：在【属性/方法列表】中，图标表示方法，图标表示属性。

（5）在【属性/方法列表】中找到 Cells 属性，双击该属性即可将其填写到代码窗口。接着输入相应的代码即可，如图 8-13 所示。

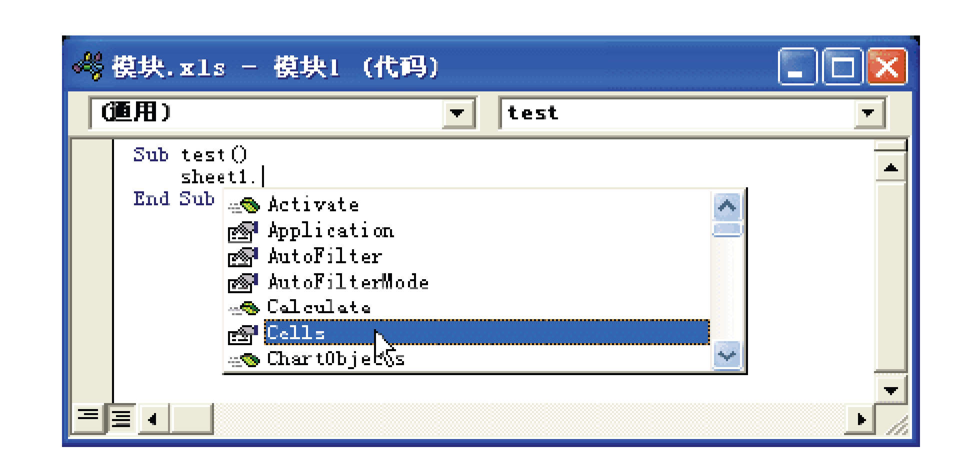


图 8-12　属性/方法列表

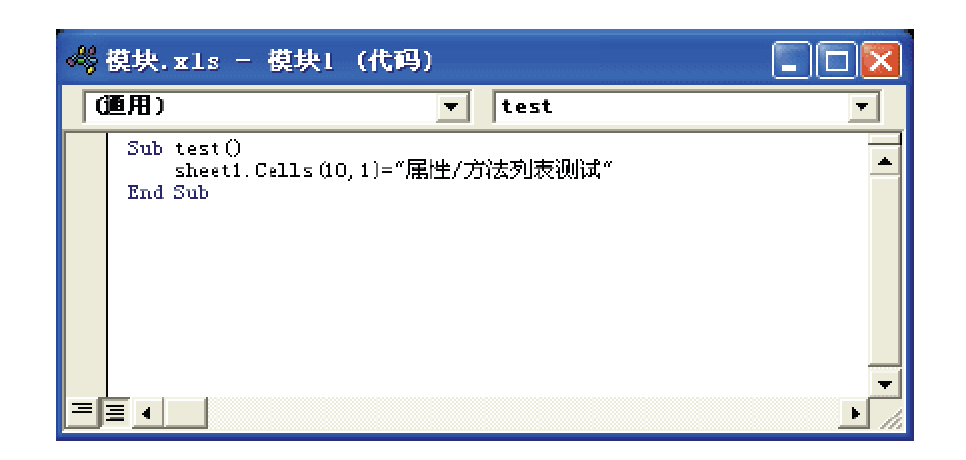


图 8-13　输入代码

打开【属性/方法列表】后，可用鼠标拖动右侧的滚动条查找所需要的属性或方法，找到后双击该项目，可将其插入到代码中。也可不理会弹出的列表，在句点后面接着输入属性或方法的前几个字母，当 VBE 突出显示需要的项目时，按下 Enter 键可插入该项目到代码中并开始新的一行（可以按 Tab 键插入项目，并在当前行继续输入代码）。

按 Esc 键可关闭列表框，并且不插入任何内容到代码窗口中。当按 Esc 键取消了弹出的列表框后，对同样的对象 VBE 将不会再弹出该列表框，可使用以下方法再次显示该列表框。

- 单击【编辑】工具栏中的【属性/方法列表】按钮。
- 按组合键 Ctrl+J。
- 右击代码窗口中的对象代码，在弹出的快捷菜单中选择【属性/方法列表】命令。

8.4.3　常数列表

在程序中，经常需要使用到一些固定的值（如一个整数、字符串等），而这些数值很抽象。例如，用 0 表示星期天、1 表示星期一，这时程序中的 0 和 1 可能有多种解释，如

果用 Sunday 和 Monday 来表示，则其意义将非常明确，因此，可在程序中定义 Sunday=0 和 Monday=1 两个常数，以增强程序的可读性。

VBA 中还提供了很多预定义的常数。在编写程序时，手工输入这些常数容易出错，通过 VBE 提供的常数列表可快速、准确地输入这些常数。

提示：有关常数的相关概念、定义参见本书第 4 章的相关介绍。

在 Excel 中，通过 Window 对象的 View 属性，可设置在窗口中显示的视图。设置该视图可分为以下 3 种方式，这三种方式定义为 XlWindowView 枚举类型。

- xlNormalView：值为 1，表示普通视图方式。
- xlPageBreakPreview：值为 2，表示分页预览视图。
- xlPageLayoutView：值为 3，表示页面视图方式。

这些常数名很长，开发人员不容易记忆。这时，可使用 VBE 提供的常数列表来帮助用户输入，具体步骤如下：

（1）在代码窗口中输入以下代码：

```
ActiveWindow.View =
```

（2）输入等号（=）后，VBE 可能会自动弹出如图 8-14 所示的常数列表，供用户选择需要设置的值。

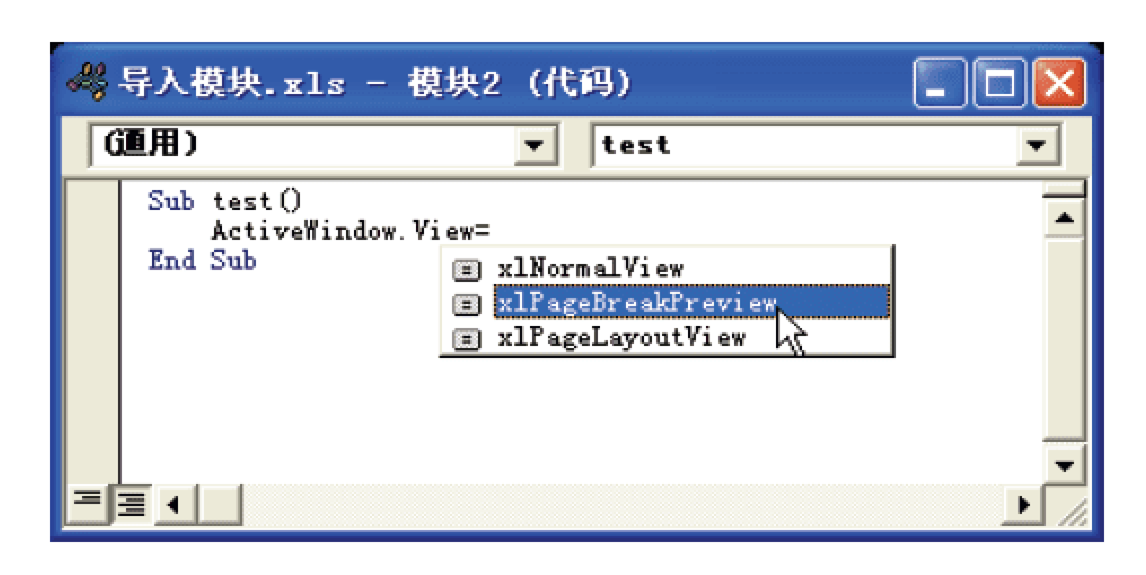


图 8-14　常数列表

（3）双击 xlPageBreakPreview 选项即可将其输入到代码中。

按 Esc 键可关闭该列表框，通过单击工具栏中的【常数列表】按钮（或按 Ctrl+Shift+J 组合键）可以再次显示出常数列表。

8.4.4　快速信息

当在代码窗口中选择了 VBA 指令、函数、方法、过程名或常数后，再单击【编辑】工具栏上的【快速信息】按钮（或按 Ctrl+I 快捷键），VBA 将会显示所选项目的语法或常数的值。使用快速信息查看 VBA 关键字的方法如下：

（1）在如图 8-15 所示代码中，鼠标单击 xlPageBreakPreview（或选中该常数）。

（2）按 Ctrl+I 快捷键，将显示插入点所在常数的值，如图 8-15 所示（本例显示出该常数的值）。



图 8-15　快速信息

8.4.5　参数信息

在 VBA 的函数中，通常每个参数的顺序是一定的。调用函数时按参数的顺序输入参数即可。

在 VBA 中支持命名参数。所谓命名参数是指参数在对象库中预先定义了其名称。对每个参数，不必局限于语法所规定的特定顺序来提供值，而是可以按任何顺序用命名参数分配值。例如，假设 MsgBox 函数的语法格式如下：

```
MsgBox(prompt[, buttons] [, title] [, helpfile, context])
```

调用该函数时可使用以下格式（按参数的顺序赋值）：

```
MsgBox "测试参数信息",,"测试"
```

以上代码省略了按钮（buttons）参数，因为是按参数顺序赋值，所以需要连续输入两个逗号。如果使用命名参数调用，则可使用以下格式：

```
MsgBox prompt:="测试参数信息", title:="测试"
```

命名的参数不必按语法中安排的正规顺序出现，以上语句写成以下形式也可得到同样的结果：

```
MsgBox title:="测试", prompt:="测试参数信息"
```

在 VBE 中输入 VBA 函数时，如果该函数需要参数，在输入函数名和左括号后，在光标下将显示一个提示框。该提示框显示函数需要的参数，在参数信息中列出了参数的顺序以及每个参数的数据类型。

例如：在【模块 1】窗口中输入一个过程 test，在过程中输入内置函数 MsgBox，再输入空格键（或括号）后将弹出如图 8-16 所示的参数信息提示框，其中列出了 MsgBox 函数中各参数的名称及数据类型等内容，可以按照参数提示信息输入函数的各个参数。

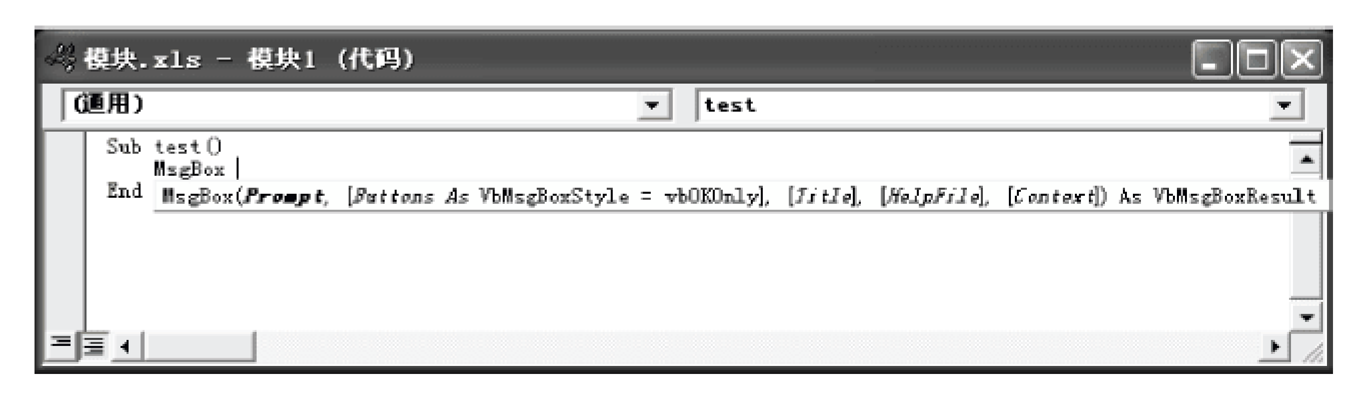


图 8-16　参数信息

8.4.6 自动完成关键字

VBA 中很多对象、方法或属性的关键字很长，输入时容易出错。VBE 为开发人员提供了自动完成关键字的功能，可帮助开发人员快速、准确地输入这些关键字。

在代码窗口中，当输入一个关键字的前几个字母，然后单击【编辑】工具栏上的【自动完成关键字】按钮（或按 Ctrl+空格组合键）后，VBE 会自动输入该关键字的剩余字母。如果输入的前几个字母可适用于多个关键字，VBE 将弹出一个列表框供用户选择。

例如，下面的操作步骤可使用户自动完成关键字的输入：

（1）启动 Excel 2007，或新建一个工作簿。

（2）按快捷菜单 Alt+F11 进入 VBE 开发环境。

（3）单击主菜单【插入】|【模块】命令向工程中插入一个模块。

（4）在新插入的模块中输入一个过程 test，在过程中输入对象 worksheets 的前几个字母 work，如图 8-17 所示。

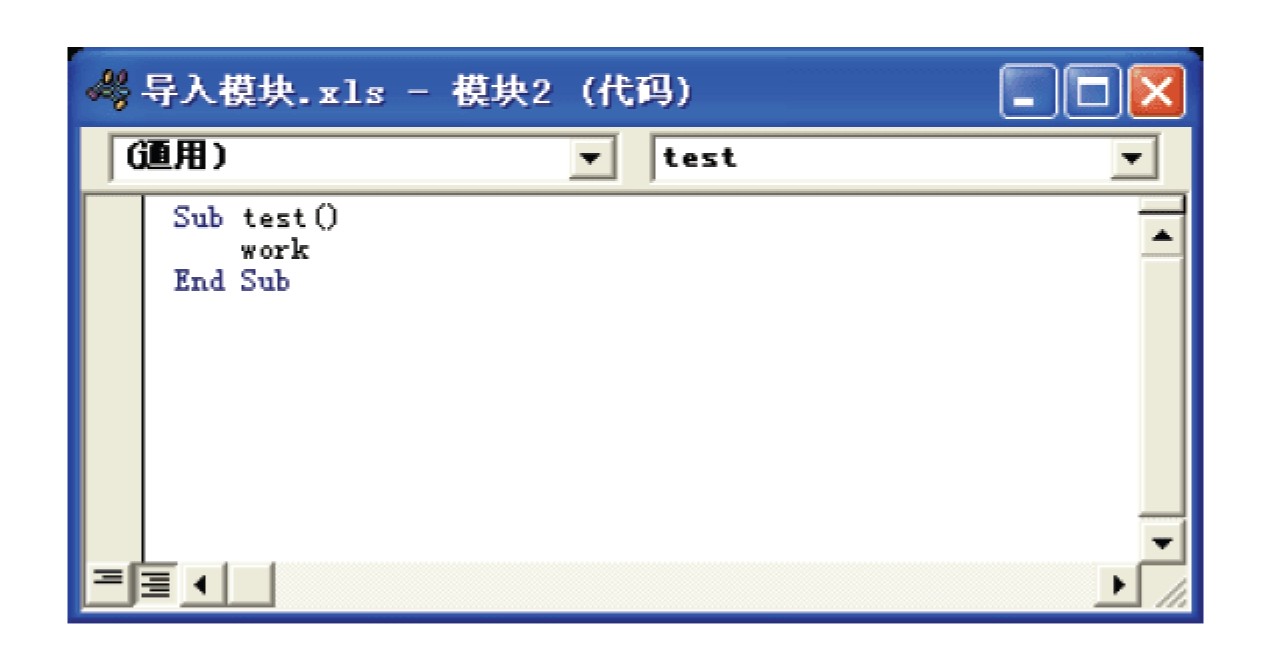


图 8-17 输入关键字的前几个字母

（5）单击主菜单【编辑】|【自动完成关键字】命令，将显示如图 8-18 所示的列表框，其中列出了与输入字母相匹配的关键字。

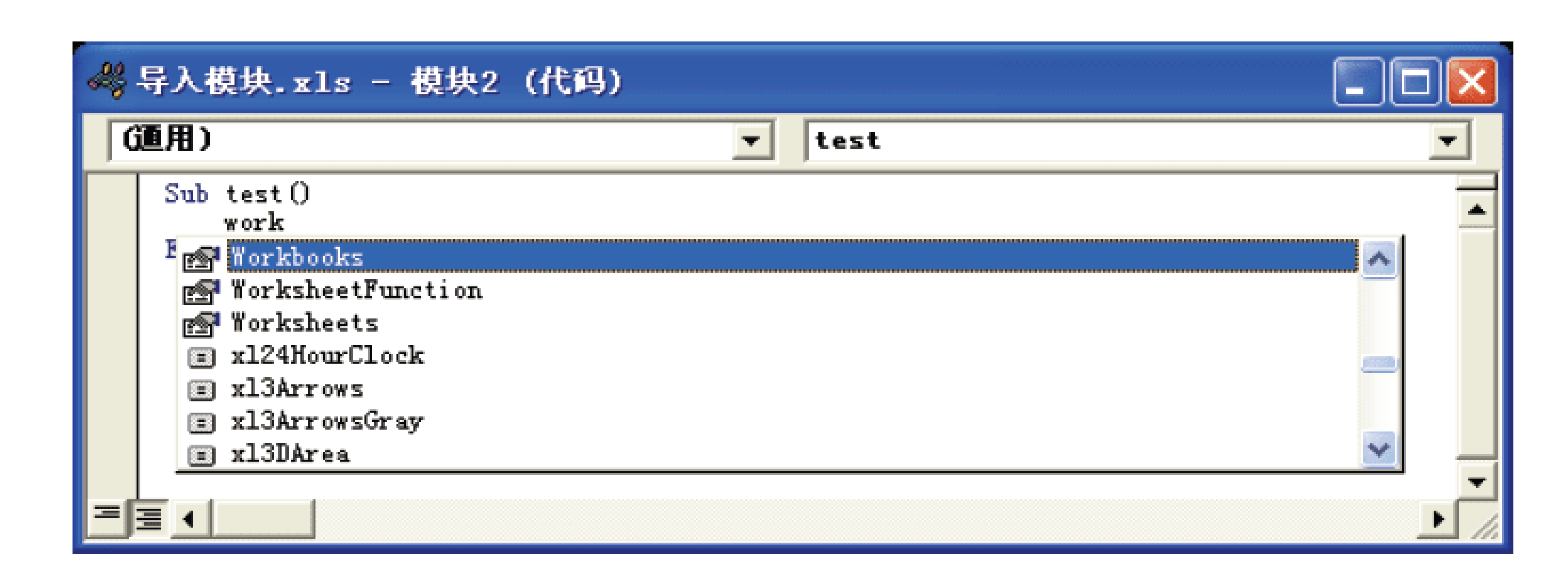


图 8-18 显示关键字列表

（6）在列表中双击 Worksheets，可将该关键字填充到代码窗口。

注意：VBE 中环境中按快捷键 Ctrl+空格也可调用【自动完成关键字】命令。但在 Windows XP 中文版中，默认情况下，Ctrl+空格为打开/关闭中文输入法的快捷键，因此，该快捷键将失效。要在 VBE 中使用该快捷键功能，需要修改 XP 的输入法快捷键。

（7）接着输入后续的代码，如图 8-19 所示。



图 8-19　输入代码

第 9 章　处理字符串和日期

在 Excel 工作表中，既可以保存和处理文字类型数据，也可保存和处理日期类型的数据。Excel 提供了很多处理文字或日期的函数，可直接在工作表中使用。VBA 为 VB 的子集，所以也可使用 VB 中的大部分函数处理文字和日期。

9.1　了解处理字符串

字符串处理是程序设计中最常见的操作，一般来说，掌握对字符串的处理也是开始学习一种新语言的基础，对后续的深入学习是非常重要的。本节首先简单介绍字符串的存储方式。

9.1.1　字符串的存储

VBA 字符串就是字节的集合，为了使 VBA 方便地处理字符串，每个字符串还保存了字符串的长度信息。

1．定义字符串变量

VBA 提供了两类字符串：定长字符串和变长字符串。定长字符串的声明方式如下：

```
Dim str1 As String * 10
```

此时，变量 str1 只能保存 10 个字符。并且，无论是否保存有字符，变量 str1 都将占用 10 个字符的内存位置，而且除保存的字符外，其他位置以空格填充，这样字符串长度始终为 10。

相反，变长字符串的长度是可变的。按照常规方法声明的变量都是变长字符串。例如：

```
Dim str2 As String
```

这样，变量 str2 可保存任意长度的字符，字符串长度为其保存字符的实际数量。

2．字符编码

字符在内存中按一定的编码规则保存。在 Windows 操作系统中，字符可按两种编码标准保存，分别为 ANSI 标准和 Unicode 标准。

ANSI 字符集中最多只能表示 256 种字符，每个字符只占用 1 个字节的存储空间。

Unicode 标准最多可表示 65 536 个字符，每个字符占用 2 个字节的存储空间。

VBA 内部都是用 Unicode 标准保存字符串。这样，每个字符将占用 2 个字节空间。在使用过程中，VBA 将动态进行 ANSI 和 Unicode 之间的转换。

9.1.2　计算字符串长度

在 VBA 中处理字符串时，常常需要知道字符串的长度。使用 Len 函数可方便地获得字符串内字符的数目。其语法格式如下：

```
Len(string | varname)
```

参数 string 为任何有效的字符串表达式。也可使用变量 Varname，函数将返回变量占用内存的长度。

例如，以下代码将返回字符串变量的长度：

```
MyString = "Hello World"              ' 设置变量初值
MyLen = Len(MyString)                 ' 返回 11
```

使用 Len 函数还可获取变量或自定义数据类型所占用内存空间大小。例如，以下代码使用 Type...End 语句，Type 定义一个自定义数据类型 CustomerRecord，再使用 Len 函数获取其占用内存的字节数。

```
Type CustomerRecord                       ' 定义用户自定义的数据类型
    ID As Integer                         ' 将此定义放在常规模块中
    Name As String * 10
    Address As String * 30
End Type
Sub 获取自定义类型字节()
    Dim Customer As CustomerRecord        ' 声明变量
    mylen = Len(Customer)                 ' 返回 42
    MsgBox "自定义类型 CustomerRecord 所占用的字节为：" & mylen
End Sub
```

运行以上代码，将显示如图 9-1 所示的对话框，在对话框中显示出自定义类型所占的字节数。



图 9-1　自定义类型所占的字节数

以上自定义类型中，Integer 变量占用 2 个字节，再加上 2 个定长字符串变量的长度（10+30），共占用 42 个字节长度。

9.2 生成重复字符串

一般情况下，将字符串常数用引号括起来就可使用到 VBA 程序中。在一般的情况下，需要使用多个重复的字符组成一个字符串，例如，输出数据时，使用多个“_”符号生成一个分隔线。可使用多种方法生成该重复字符串。本节将主要介绍这几种方法。

9.2.1 用循环生成重复字符串

可使用以下代码来完成该字符串：

```
Sub 用循环生成重复字符串()
    Dim s As String
    For i = 1 To 10
        s = s & "_"
    Next
    Debug.Print s
End Sub
```

运行以上代码，在【立即窗口】列表框中将显示一条直线，如图 9-2 所示。

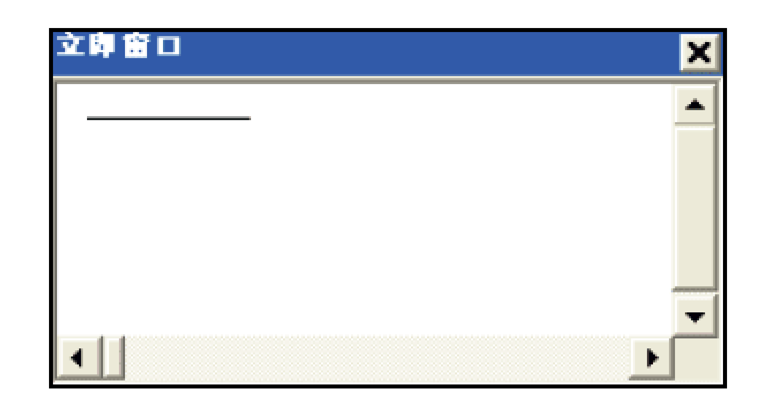


图 9-2　生成重复字符串

9.2.2 用 String 函数生成重复字符串

在 VBA 中还提供了一个 String 函数，可用来完成上面代码的功能，生成一个包含指定长度重复字符的字符串。其语法格式如下：

```
String(number, character)
```

其中各参数的意义如下所示。

- number：为返回的字符串长度。
- character：为指定字符的字符码或字符串表达式，其第一个字符将用于建立返回的字符串。

如果指定 character 的数值大于 255，String 会按下面的公式将其转换为有效的字符码：

```
character Mod 256
```

例如，使用以下代码可生成一个指定长度，且只含单一字符的字符串：

```
Sub 生成重复字符()
    Debug.Print String(5, "*")         ' 返回 "*****"
    Debug.Print String(5, 42)          ' 返回 "*****"
    Debug.Print String(10, "ABC")      ' 返回 "AAAAAAAAAA"
End Sub
```

运行以上代码，在【立即窗口】的列表框中将输出各重复字符，如图 9-3 所示。

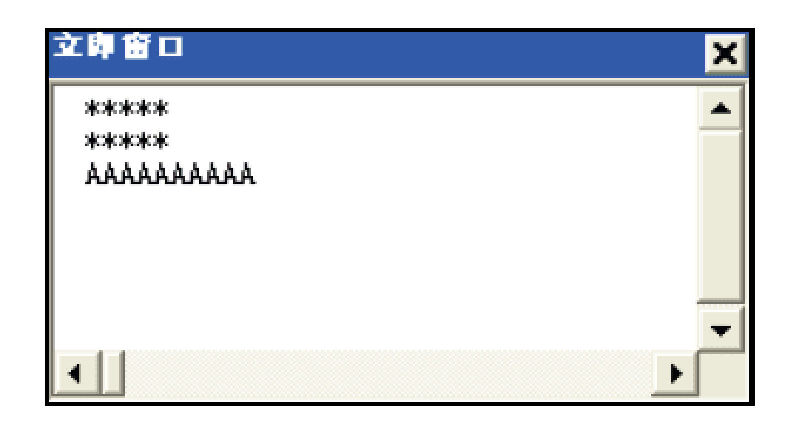


图 9-3　生成重复字符

使用 String 函数创建一条分隔线，可使用以下语句：

```
Debug.Print String(10, "_")
```

由该语句就可完成多条循环语句的功能。

9.2.3　使用 Space 函数生成重复空格

使用 Space 函数可生成指定数目空格的字符串。其语法格式如下：

```
Space(number)
```

参数 number 为字符串中想要的空格数。

Space 函数在格式输出或清除固定长度字符串数据时很有用。例如，以下代码用 Space 函数来生成一个字符串，字符串的内容为 5 个空格，使用该字符串来分隔单词。

```
Sub 生成重复空格()
    Debug.Print "12345678901234567890"
    Debug.Print "Hello" & Space(5) & "Excel VBA"     '将 5 个空格插入两个字符串
                                                     中间
End Sub
```

运行以上代码，在【立即窗口】列表框中输出的内容如图 9-4 所示。

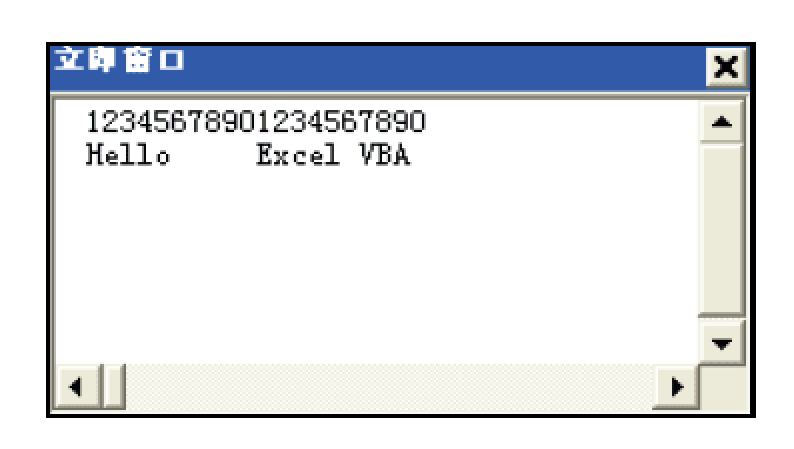


图 9-4　生成重复空格

为了观察生成的空格数量，以上代码特别输出了一串数字，用来和下方输出的字符串对应。

9.3　变换字符串

字符串的转换包括英文字母大小写的转换、字母和编码的转换两种方式。VBA 提供了多个函数来完成字符串的转换。

9.3.1　大小写字母转换——Lcase 函数和 Ucase 函数

使用 Lcase 函数可将字符串中的大写字母转换为小写字母。使用 Ucase 函数可将字符串中的小写字母转换为大写字母。

例如，以下代码将对字符串中的大小写字母进行转换：

```
Sub 大小写字母转换()
    Dim str1, strLower, strUpper
    str1 = "Hello Excel 2007 VBA"         ' 要输送的字符串
    strLower = LCase(str1)                ' 返回" hello excel 2007 vba"
    strUpper = UCase(str1)                ' 返回" HELLO EXCEL 2007 VBA"
    Debug.Print strLower
    Debug.Print strUpper
End Sub
```

运行以上代码，在【立即窗口】列表框中输出的内容如图 9-5 所示。

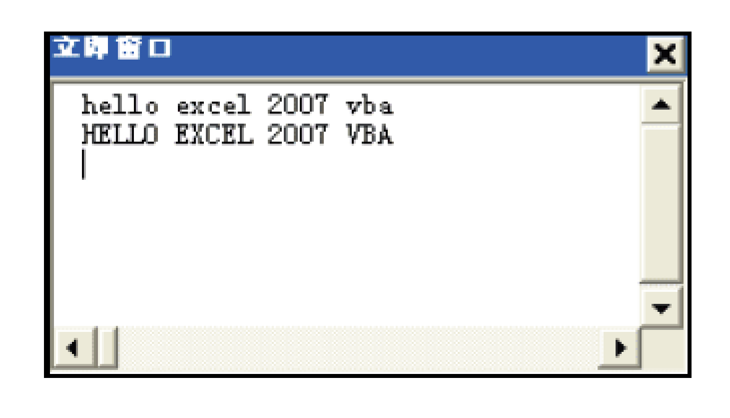


图 9-5　大小写转换

9.3.2　字符转换——StrConv 函数

StrConv 函数的功能很强大，既可转换字母大小写，还可将字符串首字母转换为大写。该函数的语法格式如下：

```
StrConv(string, conversion, LCID)
```

各参数的含义如下所述。

- ❑ string：转换的字符串表达式。
- ❑ conversion：为一个整型值，其值的和决定转换的类型。

- LCID：该参数可省略，一般不使用。

参数 conversion 决定转换的类型，可设置 9 个值，下面列出的为其中常用的 3 个值。

- vbUpperCase：值为 1，将字符串文字转成大写。
- vbLowerCase：值为 2，将字符串文字转成小写。
- vbProperCase：值为 3，将字符串中每个字的开头字母转成大写。

例如，以下代码将对字符串中的大小写字母进行转换：

```
Sub 字符转换()
    Dim str1, strLower, strUpper, strProper
    str1 = "Hello Excel 2007 VBA"                ' 要输送的字符串
    strLower = StrConv(str1, vbLowerCase)    ' 返回" hello excel 2007 vba"
    strUpper = StrConv(str1, vbUpperCase)    ' 返回" HELLO EXCEL 2007 VBA "
    strProper = StrConv(str1, vbProperCase) ' 返回" Hello excel 2007 vba "
    Debug.Print strLower
    Debug.Print strUpper
    Debug.Print strProper
End Sub
```

运行以上代码，在【立即窗口】列表框中输出的内容如图 9-6 所示。

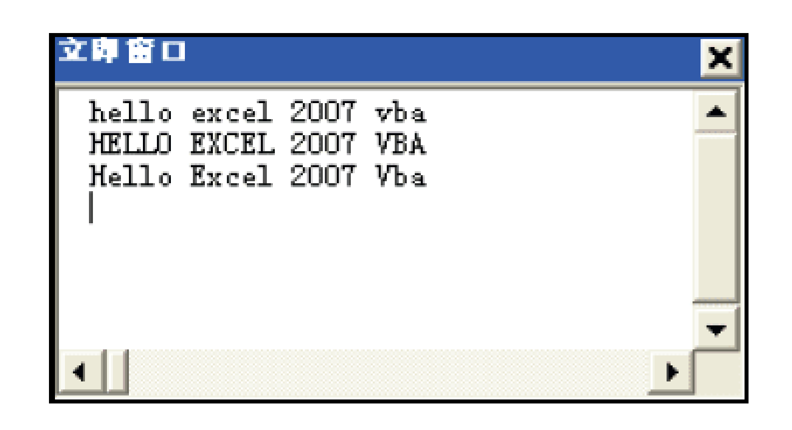


图 9-6　字符转换

9.3.3　查询字符编码——Asc 函数

Asc 函数可返回字符串中首字母的字符代码。Asc 函数的语法格式如下：

```
Asc(string)
```

参数 string 可以是任何有效的字符串表达式。如果 string 中没有包含任何字符，则会产生错误。

例如，以下代码使用 Asc 函数返回字符串首字母的字符值（ASCII 值）。

```
Sub 查询字符编码()
    Debug.Print "字符 A 的编码: " & Asc("A")              ' 返回 65
    Debug.Print "字符 a 的编码: " & Asc("a")              ' 返回 97
    Debug.Print "字符串 Excel 的编码: " & Asc("Excel")    ' 返回 69
End Sub
```

运行以上代码，在【立即窗口】列表框中输出的内容如图 9-7 所示。

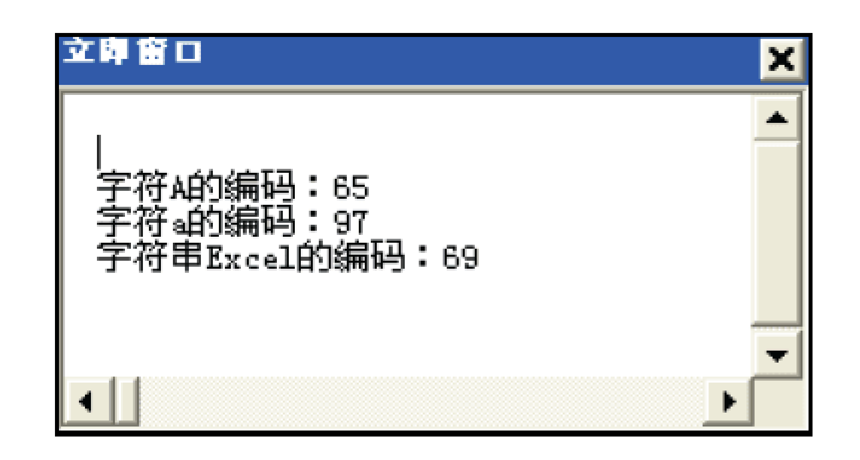


图 9-7　查询字符编码

9.3.4　生成字符——Chr 函数

使用 Chr 函数可将字符代码转换为对应的字符，其语法格式如下：

```
Chr(charcode)
```

参数 charcode 是一个用来识别某字符的代码。0～31 之间的数字与标准的非打印 ASCII 代码相同。例如，Chr(10)可以返回换行字符。charcode 的正常范围为 0～255。

例如，以下代码使用 Chr 函数来返回指定字符码所代表的字符。

```
Sub 生成字符()
    Debug.Print "编码 65 对应的字母：" & Chr(65)    ' 返回 A
    Debug.Print "编码 97 对应的字母：" & Chr(97)    ' 返回 a
    Debug.Print "编码 69 对应的字母：" & Chr(69)    ' 返回 E
    Debug.Print "编码 37 对应的字母：" & Chr(37)    ' 返回 %
End Sub
```

运行以上代码，在【立即窗口】列表框中输出的内容如图 9-8 所示。

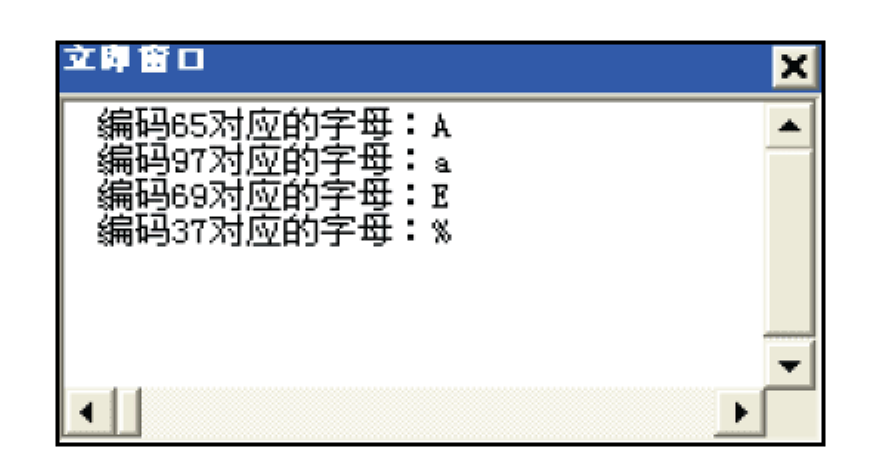


图 9-8　生成字符

9.4　比较字符串

VBA 提示了 3 种比较字符串的方式，分别为：

- ❑ 使用比较运算符（如>、<、=、>=、<=等）；
- ❑ 使用 Like 运算符；
- ❑ 使用 StrComp 函数。

9.4.1　使用比较运算符

可以使用简单的逻辑运算符进行两个字符串的比较，例如：

```
If "excel" > "Excel" Then
```

VBA 将两个比较字符串进行逐字符比较，比较的结果将根据 Option Compare 语句的设置而有所不同。

Option Compare 语句用于声明字符串比较时所用的默认比较方法。其语法格式如下：

```
Option Compare {Binary | Text | Database}
```

Option Compare 语句为模块指定字符串比较的方法（Binary、Text 或 Database）。如果模块中没有 Option Compare 语句，则默认的文本比较方法是 Binary。

- ❑ Option Compare Binary 是根据字符的内部二进制表示而导出的一种排序顺序来进行字符串比较。在 Windows 中，排序顺序由代码页确定。典型的二进制排序顺序如下例所示：

```
A < B < E < Z < a < b < e < z
```

- ❑ Option Compare Text 根据由系统区域确定的一种不区分大小写的文本排序级别来进行字符串比较。当使用 Option Compare Text 对相同字符排序时，会产生下述文本排序级别：

```
(A=a) < (B=b) < (E=e) < (Z=z)
```

- ❑ Option Compare Database 只能在 Access 中使用。当需要字符串比较时，将根据数据库的区域 ID 确定的排序级别进行比较。

注意：*Option Compare 语句必须写在模块的所有过程之前。*

9.4.2　使用 Like 运算符

还可使用 Like 运算符用来比较两个字符串。其语法格式如下：

```
result = string Like pattern
```

各参数的含义分别如下。

- ❑ string：可为任何字符串表达式。
- ❑ pattern：任何字符串表达式，可使用通配符、字符表和字符范围，如表 9-1 所示。

表 9-1　模式匹配

pattern 中的字符	符合 string 中的字符
?	任何单一字符
*	零个或多个字符

续表

pattern 中的字符	符合 string 中的字符
#	任何一个数字（0~9）
[charlist]	charlist 中的任何单一字符
[!charlist]	不在 charlist 中的任何单一字符

如果 string 与 pattern 匹配，则 result 为 True；如果不匹配，则 result 为 False。

Like 运算符的特性随着 Option Compare 语句而不同。

例如，以下代码使用 Like 运算符做字符串的方式比较。

```
Sub 使用like比较字符串()
    Debug.Print """aBBBa"" Like ""a*a""的结果为："; "aBBBa" Like "a*a"
    Debug.Print """F"" Like ""[A-Z]""的结果为："; "F" Like "[A-Z]"
    Debug.Print """F"" Like ""[!A-Z]""的结果为："; "F" Like "[!A-Z]"
    Debug.Print """a2a"" Like ""a#a""的结果为："; "a2a" Like "a#a"
    Debug.Print """aM5b"" Like ""a[L-P]#[!c-e]""的结果为："; "aM5b" Like
    "a[L-P]#[!c-e]"
    Debug.Print """BAT123khg"" Like ""B?T*""的结果为："; "BAT123khg" Like
    "B?T*"
    Debug.Print """CAT123khg"" Like ""B?T*""的结果为："; "CAT123khg" Like
    "B?T*"
End Sub
```

运行以上代码，在【立即窗口】列表框中输出的内容如图 9-9 所示。

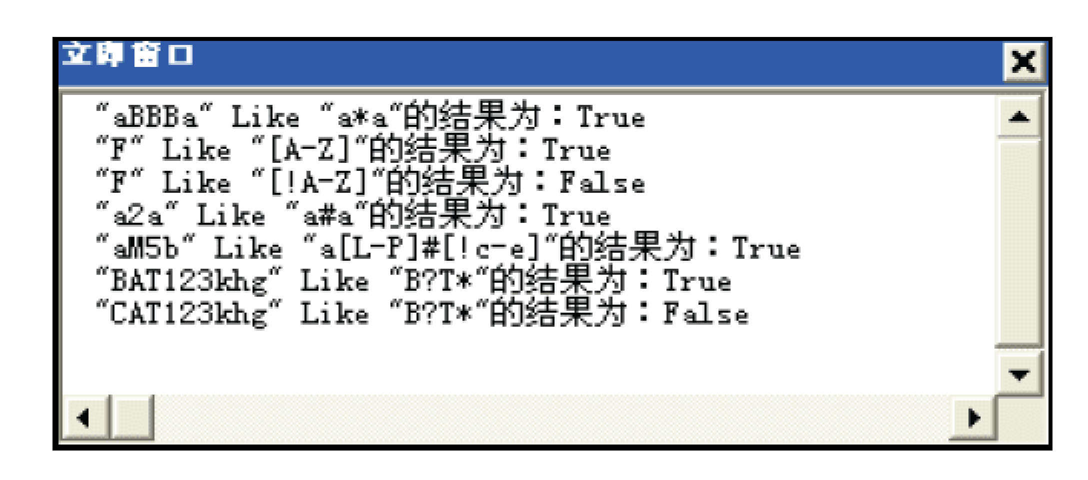


图 9-9　使用 Like 比较字符串

9.4.3　使用 StrComp 函数

使用 StrComp 函数可对两个字符串进行比较，在进行字符串比较时可设置与 Option Compare 语句不同的比较方式。该函数的语法格式如下：

```
StrComp(string1, string2[, compare])
```

以上语句中的两个文本字符串可以是任何字符串表达式，compare 参数可设为以下值之一，用来指定字符串的比较方法。

- vbUseCompareOption：值为–1，使用 Option Compare 语句设置执行比较。
- vbBinaryCompare：值为 0，执行一个二进制比较。
- vbTextCompare：值为 1，执行一个按照原文的比较。

- vbDatabaseCompare：值为 2，仅适用于 Access，执行一个基于数据库信息的比较。

StrComp 函数的返回值有以下几种情况：

- string1 小于 string2，返回值为–1；
- string1 等于 string2，返回值为 0；
- string1 大于 string2，返回值为 1；
- string1 或 string 2 为 Null，返回值为 Null。

例如，以下代码使用 StrComp 函数对字符串进行比较：

```
Sub 使用 StrComp 比较字符串()
    Dim str1, str2, MyComp
    str1 = "ABCD"
    str2 = "abcd"
    Debug.Print StrComp(str1, str2, 1)      ' 返回 0
    Debug.Print StrComp(str1, str2, 0)      ' 返回 -1
    Debug.Print StrComp(str2, str1)         ' 返回 1
End Sub
```

9.5　处理子字符串

在处理字符串时，经常需要获取字符串中的一部分内容，这部分内容称为子串。在 VBA 中可使用以下函数来处理子串。

- Left：返回字符串中左侧指定数量的字符。
- Right：返回字符串中右侧指定数量的字符。
- Mid：返回的字符串是指定位置开始，指定字符长度的字符。
- InStr：返回一字符串在另一字符串中最先出现的位置。
- Ltrim：返回去掉左侧空白的字符串。
- Rtrim：返回去掉右侧空白的字符串。
- Trim：返回去掉两侧空白的字符串。

9.5.1　取左侧子串——Left 函数

使用 Left 函数可返回指定字符串左侧的部分字符，其语法格式如下：

```
Left(string, length)
```

各参数的含义分别如下。

- string：字符串表达式。其中最左边的那些字符将被返回。
- length：为数值表达式，指出将返回多少个字符。如果为 0，返回零长度字符串("")。如果大于或等于 string 的字符数，则返回整个字符串。

例如，以下代码使用 Left 函数来得到某字符串最左边的几个字符。

```
Sub 取左侧子串()
```

```
    Dim str1 As String
    str1 = "Hello Excel 2007 VBA"    ' 定义字符串
    Debug.Print Left(str1, 1)        ' 返回"A"
    Debug.Print Left(str1, 7)        ' 返回"Hello E"
    Debug.Print Left(str1, 30)       ' 返回"Hello Excel 2007 VBA"
End Sub
```

运行以上代码，在【立即窗口】列表框中输出相应的内容，如图 9-10 所示。

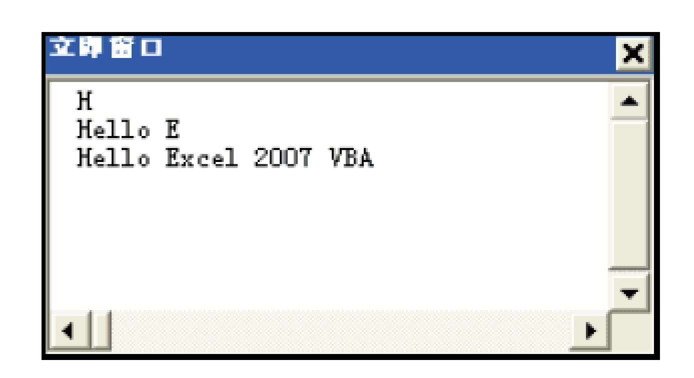


图 9-10　取左侧子串

9.5.2　取右侧子串——Rigth 函数

使用 Right 函数可返回指定字符串右侧的部分字符，其语法格式如下：

```
Right(string, length)
```

Right 函数的参数与 Left 函数的参数类似。

例如，以下代码使用 Right 函数来得到某字符串最右边的几个字符。

```
Sub 取右侧子串()
    Dim str1 As String
    str1 = "Hello Excel 2007 VBA"     ' 定义字符串
    Debug.Print Right(str1, 1)        ' 返回"A"
    Debug.Print Right(str1, 7)        ' 返回"007 VBA "
    Debug.Print Right(str1, 30)       ' 返回"Hello Excel 2007 VBA"
End Sub
```

运行以上代码，在【立即窗口】列表框中输出相应的内容，如图 9-11 所示。

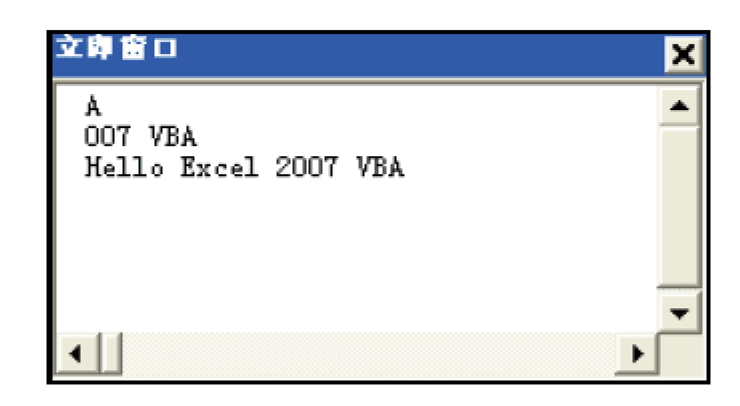


图 9-11　取右侧子串

9.5.3　获取部分子串——Mid 函数

使用 Left 函数和 Rigth 函数可分别获取字符串左侧或右侧指定数量的字符，更多的情

况下可能需要从字符串的中间某个位置开始获取一个子串，这就需使用 Mid 函数，其语法格式如下：

```
Mid(string, start[, length])
```

各参数的含义分别如下。

- string：为字符串表达式，从中返回字符。
- start：为 Long 型数据。指定从 string 中被取出子串的起始位置。如果 start 超过 string 的字符数，Mid 返回零长度字符串（""）。
- length：为 Long 型数据。指定要返回子串的字符数。如果省略或 length 超过文本的字符数（包括 start 处的字符），将返回字符串中从 start 到尾端的所有字符。该参数可省略。

例如，以下代码使用 Mid 函数取字符串中的子串：

```
Sub 获取部分子串()
    Dim str1 As String
    str1 = "Hello Excel 2007 VBA"         ' 定义字符串
    Debug.Print Mid(str1, 1, 5)           ' 返回"Hello"
    Debug.Print Mid(str1, 7, 5)           ' 返回"Excel"
    Debug.Print Mid(str1, 7)              ' 返回"Excel 2007 VBA"
End Sub
```

运行以上代码，在【立即窗口】列表框中输出相应的内容，如图 9-12 所示。

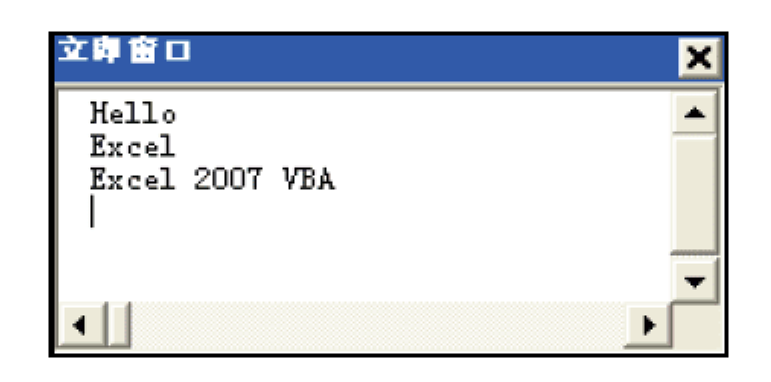


图 9-12　获取部分子串

9.5.4　删除字符串两侧空格

在 VBA 程序中，有时从 Excel 单元格或数据库中获取的数据前后会有一定数量的空格，前面有空格将影响字符串的比较运算，在输出字符串时，前后的空格也会导致输出对位不正。在进行这些处理时，可使用 Trim 函数、LTrim 函数和 RTrim 函数去掉字符串两侧的空格。这 3 个函数的语法格式相同，下面是 Trim 函数的语法格式：

```
Trim(string)
```

参数 string 为需要处理的字符串。

例如，以下代码使用这 3 个函数分别删除字符串首尾的空格：

```
Sub 删除字符串两侧空格()
    Dim str1 As String, str2 As String
```

```
    str2 = "end"                                      '用来定位属于空格的位置
    str1 = "  Hello Excel 2007 VBA  "                 ' 定义字符串(首尾各有两个空格)
    Debug.Print "123456789012345678901234567890"      '显示坐标
    Debug.Print Trim(str1); str2
    Debug.Print LTrim(str1); str2
    Debug.Print RTrim(str1); str2
End Sub
```

运行以上代码，在【立即窗口】列表框中输出相应的内容，如图 9-13 所示。为了观察两侧空格的输出情况，在【立即窗口】列表框首行显示了一个输出字符位置号。在输出每个字符串后，紧接着该字符串输入一个“end”，以观察字符串右侧的空格是否被删除。

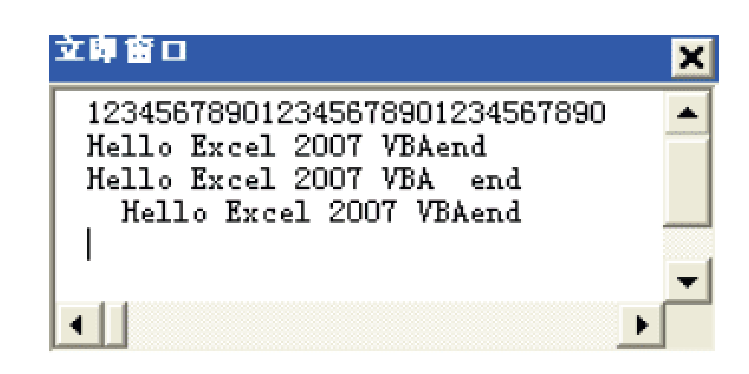


图 9-13　删除字符串两侧空格

9.5.5　查找子串位置——InStr 函数

在有的情况下，用户不知道子串在字符串中的具体位置，这时可使用 InStr 函数查找具体的位置，再使用 Mid 函数取出该子串。

InStr 函数可指定一个字符串在另一字符串中最先出现的位置。其语法格式如下：

```
InStr([start, ]string1, string2[, compare])
```

各参数的含义分别如下。

- start：为数值表达式，设置每次搜索的起点。如果省略，将从第一个字符的位置开始搜索。
- string1：接受搜索的字符串表达式。
- string2：搜索的字符串表达式，即子串。
- compare：指定字符串比较规则。如果省略 compare，Option Compare 的设置将决定比较的类型。

例如，以下代码使用 InStr 函数搜索字母“e”的位置：

```
Sub 搜索子串位置()
    Dim str1 As String
    str1 = "Hello Excel 2007 VBA"
    Debug.Print InStr(str1, "e")
    Debug.Print InStr(3, str1, "e")
    Debug.Print InStr(3, str1, "e", vbTextCompare)
End Sub
```

运行以上代码，在【立即窗口】列表框中输出相应的内容，如图 9-14 所示。



图 9-14　查找子串位置

在以上代码中，语句：

```
InStr(3, str1, "e")
```

从第 3 个字符开始，以系统默认的方式搜索字母“e”的位置，默认情况下使用 vbBinaryCompare 方式对字符进行比较，因此，搜索时将跳过大写字母“E”，找到第 10 个字符处的小写字母“e”。

而以下语句：

```
InStr(3, str1, "e", vbTextCompare)
```

使用 vbTextCompare 方式对字符比较，这时小写字母“e”和大写字母“E”是一样的，因此，返回的值为 7。

9.6　处理日期时间数据

在信息处理系统中，日期时间信息是最常用的数据之一。因此，了解日期时间类型数据的存储、处理过程，是开发 Excel 应用程序必备的知识。

9.6.1　日期时间数据的保存

在 Excel 的单元格中，可按各种形式表示日期。不论以何种形式表示日期，在 Excel 内部都是以一个序列数的形式来保存日期数据。数值的整数部分表示从 1899 年 12 月 30 日算起的天数，数值的小数部分表示当天的具体时间。

例如，北京奥运会开幕式时间为 2008 年 8 月 8 日 20:00，在 Excel 中将其存放为 39668.833。表示该日期距 1899 年 12 月 30 日为 39668 天，而时间 20:00 则为当天的 20/24。

对 VBA 用户来说，一般不用理会数据内部的保存方式，VBA 能自动进行浮点数与显示的日期格式间的转换。

9.6.2　获取和设置日期

在 Windows 等操作系统中，用户可查看和设置计算机系统当前的日期和时间。在 VBA 程序中，也提供了相应的函数和语句来获取和设置系统日期。

1. 获取当前日期和时间

使用 Now 函数可返回计算机系统的日期和时间。

例如，以下代码将显示如图 9-15 所示的对话框，显示当前日期和时间。

```
Sub 显示当前日期和时间()
    MsgBox "当前日期和时间为: " & Now
End Sub
```



图 9-15　当前日期和时间

2. 获取或设置系统日期

与 Now 函数类似，使用 Date 函数可获得系统当前的日期。

要设置系统日期，需使用 Date 语句，其语法格式如下：

```
Date = 日期表达式
```

例如，以下代码将计算机当前日期设置为 2008-8-8。

```
Date = #8/8/2008#
```

3. 获取或设置系统时间

使用 Time 函数或获得系统当前的时间。

要设置系统时间，需使用 Time 语句，其语法格式如下：

```
Time = 时间格式的字符串/数值表达式
```

如果参数是一字符串，则 Time 语句会试着根据系统指定的时间，利用时间分隔符将其转换成一个时间。如果无法转换成一个有效的时间，则会导致错误发生。

例如，以下代码将设置计算机当前时间：

```
Time = #8:08:00 PM#
```

9.6.3 生成日期/时间数据

通过 Date、Now、Time 函数可获取系统当前的日期和时间。但在 VBA 程序中，有时需要处理将来的某一个时间。这时可使用 VBA 提供的函数来生成这种日期/时间数据。

1．用字符串生成时间

TimeValue 函数可将一个表示时间的字符串生成为一个时间值。其语法格式如下：

```
TimeValue(time)
```

参数 time 通常是一个字符串表达式，表示 0:00:00（12:00:00 A.M.）到 23:59:59（11:59:59 P.M.）之间的时刻。但是，time 也可以是表示在同一时间范围取值的任何其他表达式。

可以使用 12 小时制或 24 小时制的时间格式。例如，“2:24PM”和“14:24”均是有效的 time 表达式。

如果 time 参数包含日期信息，TimeValue 将不会返回它。

例如，以下代码使用 TimeValue 函数将字符串转换为时间。

```
MyTime = TimeValue("8:08:08 PM")
```

2．使用时、分、秒数值生成时间

除了将字符串转换为时间数据外，还要使用 TimeSerial 函数按时、分、秒数值生成时间，该函数的语法格式如下：

```
TimeSerial(hour, minute, second)
```

各参数的含义分别如下。

- ❑ hour：表示小时数。其值从 0（12:00 A.M.）到 23（11:00 P.M.），或者是一个数值表达式。
- ❑ minute：表示分钟数。
- ❑ second：表示秒数。

为了指定一个时刻，TimeSerial 的参数取值应在正常范围内：小时数应介于 0～23 之间，而分钟与秒应介于 0～59 之间。但是，当一个数值表达式表示某时刻之前或其后的时、分钟或秒数时，也可以为每个使用这个数值表达式的参数指定相对时间。

当任何一个参数的取值超出正常范围时，它会适时进位到下一个较大的时间单位。例如，如果参数 minute 设置为 75（75 分钟），则这个时间被解释成一小时零十五分钟。

例如，以下代码使用 TimeSerial 函数生成时间。

```
MyTime = TimeSerial(8, 8, 8)
```

3．生成日期数据

与生成时间数据类似，VBA 也提供了两个函数（DateSerial 和 DateValue）生成日期数据。例如，下面的代码即可生成相同的日期数据。

```
MyDate = DateValue("September 13, 1973")
MyDate = DateSerial(1973, 9, 13)
```

9.6.4 计算日期数据

1．使用加减法运算

日期数据保存为一个浮点数，因此可对日期进行加减法运算。例如，以下代码将对系统当前日期进行加减计算：

```
Sub 日期运算()
    Dim MyDate As Date
    MyDate = Date
    Debug.Print "当前日期：" & MyDate                    '当前日期
    Debug.Print "10 天后的日期：" & MyDate + 10          '返回 10 天后的日期
    Debug.Print "10 天前的日期：" & MyDate - 10          '返回 10 天前的日期
End Sub
```

运行以上代码，在【立即窗口】列表框中输出相应的内容，如图 9-16 所示。

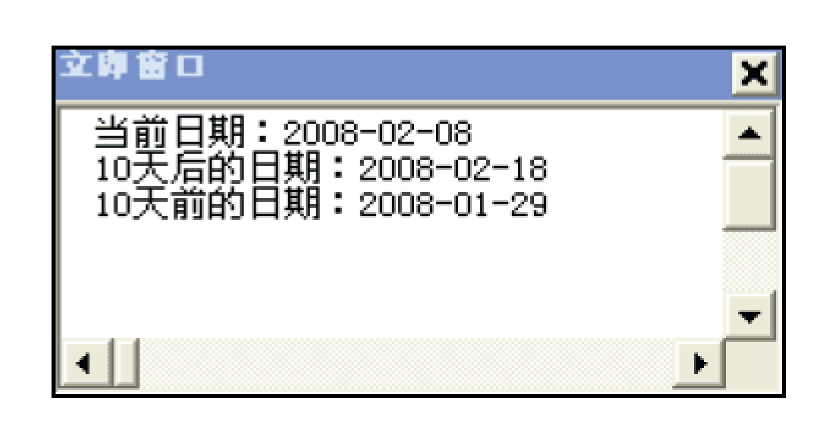


图 9-16　日期运算

2．增加日期指定部分的值

使用加减法对日期值进行运算时，只能对天数进行操作。因为日期的特殊性（月大为 31 天，月小为 30 天），只使用天数累加将得不到对应日期的下月或上月日期。例如，2008-2-8 日加 30 天则为 2008-3-9 日，若需要得到 2008-3-8 日，则不能使用这种累加天数的方法。

这时，可使用 DateAdd 函数对日期中的指定部分进行运算，其语法格式如下：

```
DateAdd(interval, number, date)
```

各参数的含义分别如下。

- interval：为一个字符串表达式，是所要加上去的时间间隔。各字符串表示的意义如表 9-2 所示。
- number：为一个数值表达式，是要加上的时间间隔的数目。其数值可以为正数（得到未来的日期），也可以为负数（得到过去的日期）。
- date：为一个表示日期的文字，用这一日期累加指定的时间间隔。

表 9-2　interval参数设定值

设　置	描　述	设　置	描　述
yyyy	年	q	季
m	月	y	一年的日数
d	日	w	一周的日数
ww	周	h	时
n	分钟	s	秒

可以使用 DateAdd 函数对日期加上或减去指定的时间间隔。例如，可以用 DateAdd 来计算距今天为三十天的日期；或者计算距现在为 45 分钟的时间。

为了对 date 加上“日”，可以使用“一年的日数”(“y”)，“日”(“d”)或“一周的日数”(“w”)。

使用以下代码可以将 1 月 31 日加上一个月：

```
DateAdd("m", 1, "31-Jan-2008")
```

DateAdd 函数将返回 2008 年 2 月 29 日，而不是 2008 年 2 月 31 日。

下面的代码演示 DateAdd 函数的使用方法：

```
Sub 增加日期指定部分的值()
    Dim MyDate As Date
    MyDate = "31-Jan-2008"
    Debug.Print "当前日期：" & MyDate
    Debug.Print "前 10 天的日期：" & DateAdd("d", -10, MyDate)
    Debug.Print "后 10 天的日期：" & DateAdd("d", 10, MyDate)
    Debug.Print "下月的今日：" & DateAdd("m", 1, MyDate)
    Debug.Print "下周同星期数的日期：" & DateAdd("ww", 1, MyDate)
    Debug.Print "明年的今日：" & DateAdd("yyyy", 1, MyDate)
End Sub
```

运行以上代码，在【立即窗口】列表框中输出相应的内容，如图 9-17 所示。



图 9-17　增加日期指定部分的值

3. 求两个日期的间隔值

使用 DateDiff 函数可返回两个指定日期间的时间间隔数目。其语法格式如下：

```
DateDiff(interval, date1, date2[, firstdayofweek[, firstweekofyear]])
```

各参数的含义分别如下。

- ❑ interval：为字符串表达式，表示用来计算 date1 和 date2 的时间差的时间间隔，其

设置值如表 9-2 所示。

- date1、date2：为计算中要用到的两个日期。
- firstdayofweek：指定一个星期的第一天的常数。如果未予指定，则以星期日为第一天。该参数可省略。
- firstweekofyear：指定一年的第一周的常数。如果未予指定，则以包含 1 月 1 日的星期为第一周。该参数可省略。

DateDiff 函数可用来决定两个日期之间所指定的时间间隔数目。例如，可以使用 DateDiff 来计算两个日期之间相隔几日，或计算从今天起到年底还有多少个星期。

如果 date1 比 date2 来得晚，则 DateDiff 函数的返回值为负数。

例如，以下代码可求出今日距北京奥运会开幕的日期：

```
Sub 距奥运会开幕日期()
    Dim oDate As Date, today As Date
    Dim t1 As Integer, t2 As Integer
    oDate = #8/8/2008#
    today = Date
    t1 = DateDiff("d", today, oDate)
    t2 = DateDiff("ww", today, oDate)
    MsgBox "今日距北京奥运会开幕还有：" & t1 & "天（" & t2 & "周）！"
End Sub
```

运行以上代码，将显示如图 9-18 所示的对话框，在该对话框中显示出了计算的日期。



图 9-18　日期间隔值

9.6.5　使用计时器

在 VBA 中，没有专门的计时器。可使用 Timer 函数来模拟计时器的功能。Timer 函数将返回一个 Single 类型的值，代表从午夜开始到现在经过的秒数。

例如，以下代码使用 Timer 函数来暂停应用程序。同时用 DoEvents 在暂停期间将控制让给其他进程。

```
Sub 暂停程序的执行()
    Dim PauseTime, Start, Finish, TotalTime
    If (MsgBox("是否暂停程序 5 秒钟", vbQuestion + vbYesNo)) = vbYes Then
        PauseTime = 5                           ' 设置暂停时间
        Start = Timer                           ' 设置开始暂停的时刻
        Do While Timer < Start + PauseTime
            DoEvents                            ' 将控制让给其他程序
        Loop
        Finish = Timer                          ' 设置结束时刻
```

```
        TotalTime = Finish - Start            ' 计算总时间
        MsgBox "程序暂停时间为: " & TotalTime & " 秒! "
    Else
        End
    End If
End Sub
```

运行以上代码，首先将显示如图 9-19 左图所示的对话框，用户单击【是】按钮，程序将暂停 5 秒，此时用户可进行其他操作。5 秒钟后，将显示如图 9-19 右图所示的提示对话框。





图 9-19　程序执行提示信息

第 3 部分　掌握 Excel 对象模型

对象模型用来描述对象之间的关系。使用 VBA 在 Excel 环境下开发应用程序，实际就是访问和控制 Excel 各对象的过程。使用 VBA 在 Excel 中进行编程时，必须先了解 Excel 的对象模型。

Excel 2007 有 200 多个对象，如果这些对象之间没有任何逻辑联系，那么这些对象将非常难于掌握和使用。事实上，在 Excel 中所有对象都处于一个完整的体系中，每个对象都不是孤立的。

本部分共 6 章，详细介绍了 Excel 中常用对象的使用方法。

第 10 章　Excel 对象概述

面向对象程序设计方法是一种非常实用的软件开发方法，其以客观世界中的对象为中心，分析和设计思想符合人们的思维方式，分析和设计的结果与客观世界的实际比较接近，容易被人们所接受。

VBA 为面向对象的程序设计语言，将其嵌入到 Excel 中后，就可以直接访问 Excel 中的各对象。本章将介绍对象、Excel 对象模型的相关知识。

10.1　对象的概念

本节介绍对象的概念，以及与对象相关的属性、方法、事件等相关内容。

10.1.1　了解对象

在面向对象的程序中，“对象”是系统中的基本运行实体。VBA 中的对象与其他面向对象程序设计语言（如 C++等）中的对象在概念上是相同的，但在使用上有很大区别。在 C++之类的程序设计语言中，需要程序设计人员编写代码设计对象。而 Excel VBA 中，Excel 已为开发人员提供了很多的对象，开发人员可直接访问这些对象。在 VBA 中，开发人员也可自己编写类，用来创建自己的对象（与 C++程序设计语言类似）。

在 Excel 中，对象是指一组属性及这组属性上的专用操作的封装体。属性可以是一些数据，也可以是另一个对象。例如，Excel 工作簿为一个对象，它的属性有工作簿名称、保存位置、作者、工作表等，而工作表又可以是一个对象，还可以有自己的属性。每个对象都有它自己的属性值，表示该对象的状态。对象中的属性只能通过该对象所提供的操作来存取和修改。操作也称为方法或服务，它规定了对象的行为，表示对象所能提供的服务。一个对象通常由对象名、属性和操作 3 部分组成。

在 Excel 中，对象代表应用程序中的元素，例如，工作表、单元格、图表、窗体，或是一份报告。在 VBA 的代码中，在使用对象的任一方法或改变它的属性之一的值之前，必须先识别对象。

10.1.2　对象的属性

属性是一个对象的特性，决定了一个对象的外观和行为。要改变一个对象的外观和行为，可以通过改变对象的属性来实现。大多数对象属性是在对象生成时自动设置的，用户

可以在设计时通过【属性】窗口修改属性值，也可在 VBA 代码运行时通过代码改变属性。有些属性值在运行时不允许 VBA 代码进行修改，这种属性称为只读属性。

下面介绍在 VBA 代码中设置和访问属性的方法。

1．设置属性值

在 VBA 代码中，可使用以下格式设置指定属性的值。

```
对象名.属性名 = 属性值表达式
```

例如，下面的代码设置 Excel 工作表 Sheet1 的标签为“第 1 张工作表”：

```
Sub 更改工作表标签()
    Sheet1.Name = "第 1 张工作表"
End Sub
```

在以上代码中，Sheet1 为工作表对象的名称，Name 为属性名。

在 VBA 的很多情况下，对象的属性又可返回一个对象，这里可通过多个句点符号（.）来逐级引用对象的子对象，及子对象的属性。例如，有以下的代码：

```
Sub 设置单元格的值()
    Sheet1.Range("A1").Value = "测试"
End Sub
```

其中 Sheet1 为 Excel 工作表对象，Range("A1")为单元格对象 A1，Value 为属性名。

2．读取属性值

可以通过属性的返回值，来检索对象的信息。读取属性值可以用以下语法：

```
变量名 = 对象名.属性名
```

例如，下面的代码可获取工作表 Sheet1 中单元格 A1 中的值：

```
Sub 获取单元格的值()
    Dim r
    r = Sheet1.Range("A1").Value
    MsgBox r
End Sub
```

属性值也可以作为表达式的一部分，而不必将属性赋予变量。下面的代码计算工作表 Sheet1 两单元格的数据之和。

```
intSum = Sheet1.Cells(2, 1).Value + Sheet1.Cells(2, 2).Value
```

在工作表 Sheet1 的对象中又包含很多对象，其中 Cells 对象为工作表单元格的集合对象，父对象引用子对象时将通过点运算符进行。

10.1.3　对象的方法

在面向过程的程序设计中，过程和函数是程序的主要部件。而在面向对象的程序设计中，引入了称为方法的特殊过程和函数。方法的操作与过程、函数相同。不同的是，方法

只能是特定对象的一部分，即一个对象的方法不能应用到另一个对象中。

在调用方法时使用点操作符引用，如果有参数，在方法后加上参数值，参数间用空格隔开。在代码中使用方法的格式如下：

```
对象名.方法名称
```

例如，单元格区域对象为 Range，可以使用 Clear 方法清除单元格中的内容。

```
Sub 清除单元格的值()
    Sheet1.Range("A1").Clear
End Sub
```

10.1.4　对象的事件

事件是一个对象可以辨认的动作，像单击鼠标或按下某键等，并且可以写某些代码针对此动作做出的响应。用户做的动作或程序代码的结果都可能导致事件的发生，或是由系统引发。在 VBA 中，可以激发事件的用户动作包括：切换工作表、选择单元格、单击鼠标等几十种事件。当事件发生时，将执行包含在事件过程中的代码。如果用户没有定义某事件所调用的过程，当发生该事件时，就不会产生任何反应。

编写事件响应代码是在【代码编辑器】窗口中进行的，如图 10-1 所示。在【代码编辑器】窗口的右边有一个事件列表，这个事件列表从属于左边的控件对象，当左边控件列表中的控件改变后，右边的事件列表也会发生变化。

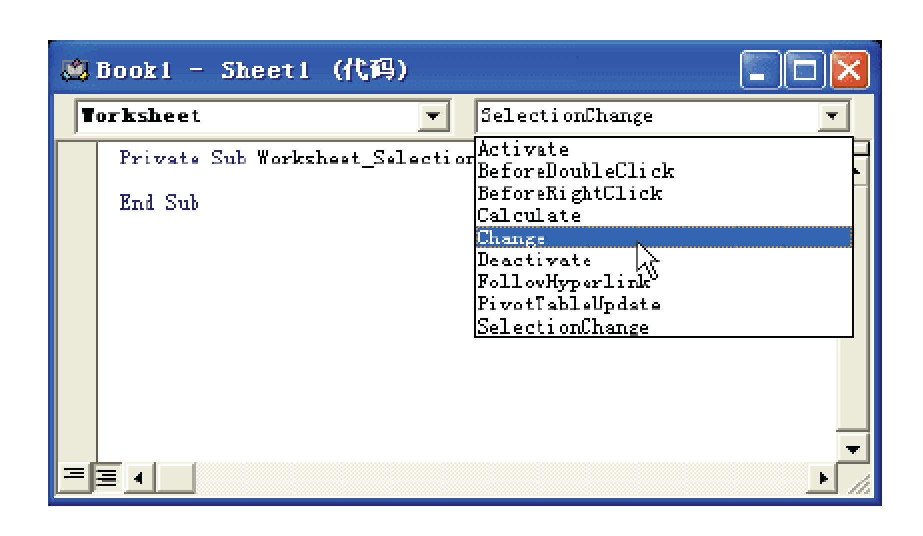


图 10-1　对象事件列表

事件过程的一般格式如下：

```
Private Sub 对象名_事件名称(参数列表)
    ……
    事件响应程序代码
    ……
End Sub
```

10.2　对象变量和对象数组

本书前面章节介绍了变量和数组的知识。在 VBA 中，变量和数组除了直接存储值，

还可以引用对象。将对象分配给变量的优点如下：

- ❑ 变量名通常要比访问对象本身所需的方法和属性的完整路径短而且容易记忆。
- ❑ 与通过所需的方法或属性来重复访问对象本身相比，使用引用对象的变量更有效。
- ❑ 在代码运行期间，可以更改变量以引用其他对象。

10.2.1　对象变量

对象变量是代表一个完整对象的变量，该变量中实际保存着具体对象的引用指针。与普通变量类似的是，使用对象变量也需要两个步骤：声明对象变量和指定对象变量到某一对象。

1．声明对象变量

可以使用 Dim 语句或其他的声明语句之一（Public，Private 或 Static）声明对象变量。引用对象的变量必须是 Variant、Object 或是一个对象的指定类型。例如，下列声明是有效的：

```
Dim MyObject                    '声明 MyObject 为 Variant 数据类型
Dim MyObject As Object          '声明 MyObject 为 Object 数据类型
Dim MyObject As Font            '声明 MyObject 为 Font 类型
```

注意：如果使用对象变量前没有声明数据类型，则对象变量默认的数据类型是 Variant 类型。

在有些情况下，只有等到程序运行时才知道对象变量引用的对象类型，这时可将对象变量声明为 Object 数据类型。使用 Object 数据类型可以创建对任何对象的一般引用。

如果知道对象变量引用的对象类型，最好将其声明为所知道的对象类型。声明指定的对象类型可以提供自动的类型检查，更快的代码生成，并增加可读性。例如，如果对象变量 MyObject 指定为 Range 对象类型，则可以用下列的语句来声明：

```
Dim MyObject As Object          '声明为一般的对象
Dim MyObject As Range           '只声明为 Range 对象
```

2．对象变量赋值

与普通变量的赋值不同，给对象变量赋值必须使用 Set 语句。例如，下面的代码将 Excel 的单元格对象 Range 赋值给对象变量 MyCell：

```
Set  MyCell = Worksheets(1).Range("C2")
```

经过以上的赋值操作后，在需引用单元格 C2 的地方，可直接使用对象变量 MyCell 来代替。例如，以下两种方式都可显示单元格 C2 保存的内容：

```
MsgBox "单元格 C2 的值为: " & MyCell.Value
MsgBox "单元格 C2 的值为: " & Worksheets(1).Range("C2").Value
```

由以上代码可看出，使用对象变量的方式可使应用程序更简练。

3．取消关联

设置一个对象变量等于 Nothing 会中断此对象变量与任何特定对象的关联，这样可预防因意外改变变量而更改对象。在关闭关联对象后，对象变量总是设置为 Nothing，所以可以检测对象变量是否指向有效的对象。例如：

```
If Not MyObject Is Nothing Then     ' 变量引用有效的对象
    ...
End If
```

当然，该检测不能绝对地决定用户是否已关闭包含对象变量所引用对象的应用程序。

注意：值为 Nothing 的对象变量也称为“空引用”。

4．使用对象变量的优点

使用对象变量可简化代码，并提高代码的执行速度。例如，如果在某段代码中需要反复使用单元格 C2，那么完整的引用代码如下：

```
Worksheets(1).Range("C2").Value = "地址"
Worksheets(1).Range("C2").Font.Name = "黑体"
Worksheets(1).Range("C2").Font.Bold = True
```

下面的代码使用对象变量来引用单元格 C2，用来完成上面同样的操作：

```
Dim MyCell As Range
Set MyCell = Worksheets(1).Range("C2")
MyCell.Value = "地址"
MyCell.Font.Name = "黑体"
MyCell.Font.Bold = True
```

这段代码更易读，并且执行效率更高。仅从这段代码上还不易觉察速度的提高，如果在一个需要重复执行几百上千次的循环中执行这段代码，则其执行效率将明显提高。

10.2.2　对象数组

如果在程序中需要处理大量相同类型的对象，这时可使用对象数组来指定这些对象。对象数组的定义与普通类型数据的数组相同，其使用与对象变量的使用类似。下面以实例形式介绍对象数组的定义和使用。

例如，如果要引用工作表中 A 列的前 10 个单元格，可使用以下代码：

```
Dim MyRange(10) As Range
For i = 1 To 10
    Set MyRange(i) = Worksheets(1).Cells(i, 1)
Next
For i = 1 To 10
```

```
    MsgBox (MyRange(i).Value)
Next
```

程序首先定义一个具有 10 个元素的对象组，接着使用一个循环将各单元格的引用赋值给对象数组，最后再使用一个循环调用该对象数组的各元素。本例只显示各单元格的值，另外可以通过对象数组中的元素 MyRange(i)来控制各单元格的字体等各种属性，例如：

```
MyRange(i).Value = 10 + I              '为单元格赋值
MyRange(i).Font.Bold = True            '设置单元格字体为粗体
```

10.3　使 用 集 合

在使用 Excel 开发应用程序时，需要大量使用到集合。通过对集合的操作可简化代码。下面介绍集合的概念及处理集合的方法。

10.3.1　集合的概念

集合是一种特定类型的对象，代表一组相同的对象。例如，一部电话是一个对象，多部电话就组成电话集合。

在 Excel 中新建一个工作簿，可以发现每个工作表都是相同的。像这样一组相似的对象就称为“集合”，集合也是对象。Excel 中使用得最频繁的集合是代表所有的工作表和图表工作表的 Sheets 集合、Workbooks 集合、Worksheets 集合及 Windows 集合。当使用集合时，可以在该集合中所有的对象上执行相同的操作。

一个集合也可能是另一个对象的属性。例如，电话中的按键集合是电话对象的属性，是单个按键对象的集合。因此，它们之间的关系是：电话集合是多部单个电话对象组成的集合，每个电话对象有一个称作按键集合的属性，而按键集合是单个按钮对象组成的集合。

10.3.2　访问集合中的对象

一个集合中包含着具有相同类型的多个对象，除了使用上面介绍的方法向集合中增加对象外，还可通过集合的属性访问集合中的单个对象。

1. Excel中的集合

Excel 中有很多系统定义的集合，如以下代码可在当前工作簿中增加一张工作表（或在工作表集合中增加一个对象）：

```
Set ws = ActiveWorkbook.Worksheets.Add
```

Worksheets 是 Worksheet（工作表）对象的集合，也是 ActiveWorkbook 对象的一个属性。在这里也可以得出集合的用处：一个工作簿可以包含任意多个工作表，而 Worksheets 集合提供了管理这些工作表的简单的方法。

Excel 中的常用的集合对象还有以下几类。

- ❑ Axes 集合：图表中所有坐标轴对象的集合；
- ❑ Charts：工作簿中图表工作表的集合；
- ❑ Sheets：工作簿中所有工作表的集合；
- ❑ Workbooks：所有打开的工作簿的集合。

2. 引用集合中的对象

使用 VBA 可以处理某个对象的整个集合，或者某集合中的一个单独的对象。引用集合中的对象的方法是：

```
集合(“对象名”) 或 集合(对象索引号)
```

引用集合中的某个对象，即引用对象名或对象索引号所代表的对象。例如，以下代码引用集合 Worksheets 中的工作表 Sheet1：

```
Worksheets("Sheet1 ")
```

如果 Sheet1 是集合中的第一个工作表对象，还可以写为以下形式：

```
Worksheets(1)
```

注意：Sheets 集合由工作簿中的所有工作表（包括图表工作表）组成。若要引用工作簿中的第一个工作表，可采用语句 Sheets(1)表示。

10.3.3 集合的方法和属性

所有的集合都有方法和属性，允许访问集合中的单个对象。其中，最重要的方法和属性有 3 个，即 Count 属性、Item 方法和 Add 方法。

1. Count属性

该属性指出在集合中有多少个单个对象。例如：

```
Dim num1
Num1=ActiveWorkbook.Worksheets.Count
```

其中，num1 是一个变量，存储 ActiveWorkbook 对象中 Worksheet 对象的数量（即工作簿中工作表的数量）。

注意：如果一个集合为空，则该集合的 Count 属性值为 0。

2. Item方法

该方法访问集合中一个特定的对象。例如：

```
Set myWorksheet=ActiveWorkbook.WorkSheets.Item(2)
```

括号中的数字表明想访问的是哪个工作表，即在 Worksheets 集合中的第 2 个 Worksheet 对象，并将它赋给 myWorksheet 变量。

也可以在括号中使用变量，例如：

```
Dim num1 as Long, LastWorksheet as Object
Num1=ActiveWorkbook.Worksheets.Count
Set LastWorksheet=ActiveWorkbook.Worksheets.Item(num1)
```

上面的代码首先用变量 numWorksheets 来存储在 Worksheets 集合中 Worksheet 对象的数量，然后访问最后的工作表。因此，如果工作簿中共有 5 个工作表，则可以指定最后项目为“5”来访问最后的工作表，所以下面的代码与上面代码最后一行等价：

```
Set LastWorksheet=ActiveWorkbook.Worksheets.Item(5)
```

3. Add方法

该方法允许向集合中添加对象。例如：

```
Set NewWorksheet=ActiveWorkbook.WorkSheets.Add("Sheet6")
```

如何使用 Add 方法取决于想要添加项目的集合对象。在多数情况下，可以为新的对象指定一个名字，例如，上面代码中的 Sheet6。

10.3.4　遍历集合中的对象

如果需要对集合中的每个对象执行相同的操作，首先需要通过集合对象的 Count 属性获取集合中包含的对象数量，然后通过循环语句对集合中的每个对象执行相同的操作。例如，以下代码将选中区域单元格中的英文字母转换为大写：

```
Dim i As Long
j = Selection.Count
For i = 1 To j
   Selection.Cells(i) = UCase(Selection.Cells(i))
Next
```

在 VBA 中提供了一个 For Each…Next 语句，该循环语句可以逐个处理集合中每个对象。使用该语句处理集合时，不必知道集合中有多个元素。其语法结构如下：

```
For Each 元素 In 集合
     [语句系列 1]
     [Exit For]
     [语句序列 2]
Next
```

使用 For Each…Next 语句可将前面例子中的代码改写为以下形式：

```
Dim MyCell As Range
For Each MyCell In Selection
   MyCell = UCase(MyCell)
Next
```

10.4　Excel 对象模型

Excel 2007 的对象很多，大到应用程序、窗体，小到一个单元格。所有这些对象都是通过一定的层次结构组织起来的。要真正掌握 Excel 2007 的 VBA 编程，不仅要求开发人员熟悉 VBA 基本语法，熟悉 Excel 2007 的对象及其层次结果，还必须通过大量的实际编程才能真正掌握 Excel 应用程序的开发方法，编写出精简高效的程序。

本节将介绍 Excel 对象模型，在本书后面的章节中将详细介绍常用 Excel 对象的使用方法。

10.4.1　Excel 对象模型简介

Excel 的对象模型是通过层次结构很有逻辑地组织在一起的，一个对象可以是其他对象的容器，可以包含其他的对象，而这些对象又包含其他的对象。位于顶层的是 Application 对象，它包含 Excel 中的其他对象，例如 Workbook 对象；一个 Workbook 对象又包含其他一些对象，例如 Worksheet 对象；而一个 Worksheet 对象又可以包含其他对象，例如 Range 对象。其层次结构如图 10-2 所示。

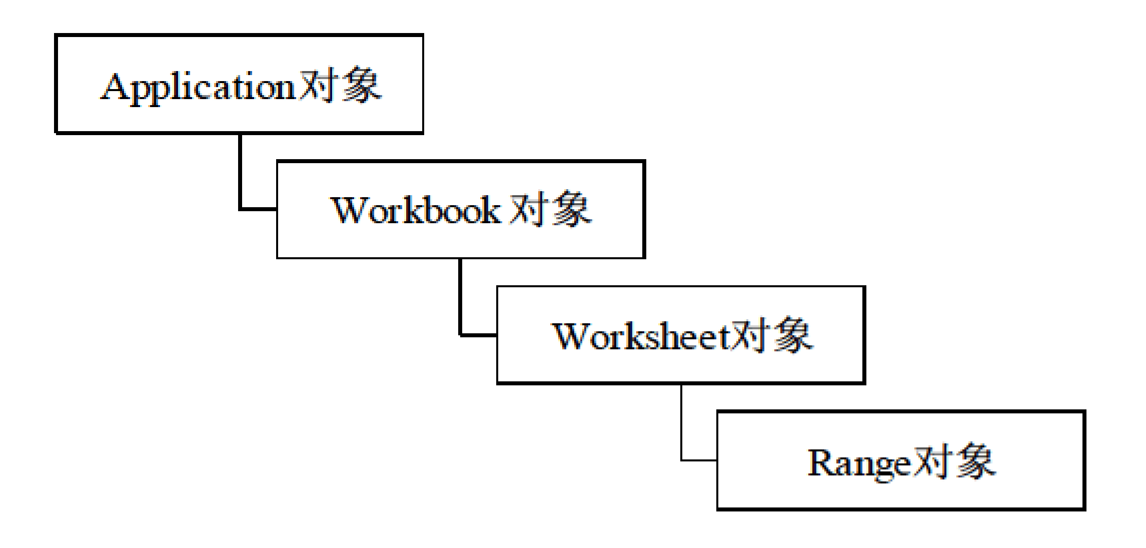


图 10-2　对象的层次结构

知道了某对象在对象模型层次结构中的位置，就可以用 VBA 代码方便地引用该对象，从而对该对象进行操作，并以特定的方式组织这些对象，使 Excel 能根据需要自动化地完成工作任务。例如，要获取单元格 A1 的值，按图 10-2 所示的引用层次，可使用以下代码：

```
Application.Workbooks(1).Worksheets(1).Range("A1").Value
```

以上代码表示引用第 1 个工作簿中第 1 张工作表的单元格 A1 的值。

要熟练掌握 Excel VBA 编程，必须理解 Excel 的对象模型。在 Excel 2007 的 VBA 中提供了 200 多种对象，在 VBA 的帮助系统中可查看每个对象的属性、方法和事件。

在 Excel 2007 的 VBA 帮助系统中没有提供 Excel 对象模型的层次结构图。如图 10-3 所示为 Excel 2003 的 VBA 帮助系统中提供的 Excel 对象模型的层次结构图，供读者参考。

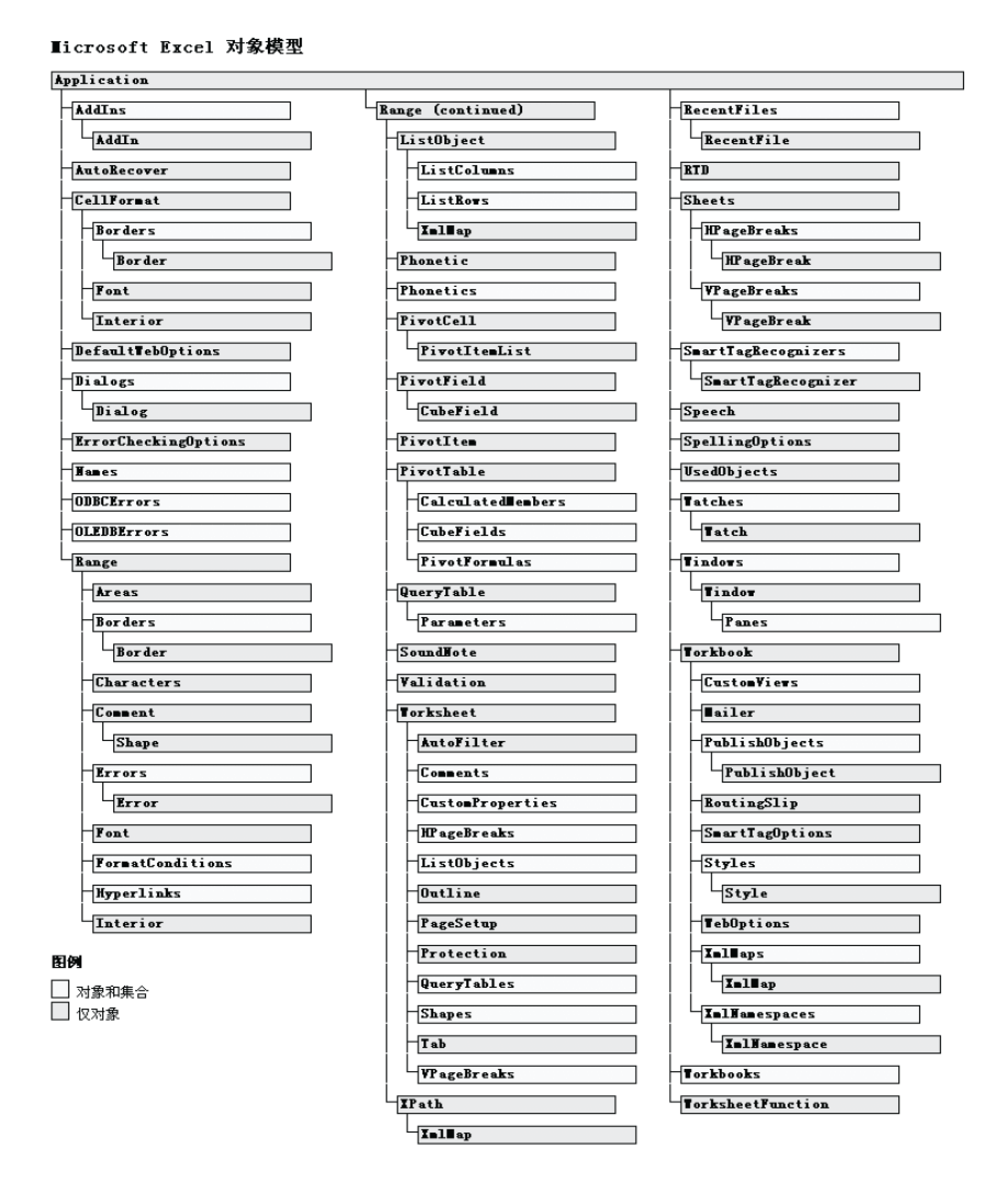


图 10-3　Excel 对象模型

由上图看出，Excel 的对象模型以 Application 为顶层对象，下属各对象按字母顺序排列，将对象、集合以不同颜色显示。其中每个对象的下属对象显示在下一层中，例如 Application 对象下的第一下属子对象为 AddIns（为一个集合对象）。其中 Range 对象的下属子对象最多，包括 Areas、Comment、Errors、Font 等子对象。

10.4.2　常用对象简介

在 Excel 2007 中，提供了二百多个对象。在使用 Excel 时，有些对象可能会经常用到，而另外一些对象则不常用。一般来说，对 Excel 应用程序本身的一些设置、对工作簿和工作表的操作、单元格和单元格区域的操作、图表的操作等是经常要涉及的。因此，在学习

Excel 的对象模型时，可以先集中研究和探讨与这些操作相关的对象、方法、属性和事件，以及它们的使用，以此来逐步加深对 Excel 对象模型的认识和理解直至全面掌握。

在开发 Excel 应用程序时，最常用的有以下对象（此处列出这些对象的主要作用，在后面章节中将详细介绍这些对象的使用方法）：

1．Application对象

Application 对象代表整个 Excel 应用程序。该对象包括：

- 应用程序范围的设置和选项。
- 返回顶级对象的方法，例如，ActiveCell 和 ActiveSheet 等。

2．Workbook对象

WorkBook 代表一个 Excel 工作簿。Workbook 对象是 Workbooks 集合的成员。Workbooks 集合包含 Excel 中当前打开的所有 Workbook 对象。

使用 Application 对象的 ThisWorkbook 属性将返回运行 VBA 代码的工作簿。在大多数情况下，该工作簿与活动工作簿是同一个。

3．Worksheet对象

Worksheet 对象代表一个工作表。Worksheet 对象是 Worksheets 集合的成员。Worksheets 集合包含某个工作簿中所有的 Worksheet 对象。Worksheet 对象也是 Sheets 集合的成员。Sheets 集合包含工作簿中所有的工作表（图表工作表和工作表）。

4．Range对象

Range 为单元格对象，是 VBA 代码中最常用的对象。Range 对象代表某一单元格、某一行、某一列、某一选定区域（该区域可包含一个或若干连续单元格区域），或者某一个三维区域。

使用以下属性和方法可返回 Range 对象。

- Range 属性，使用 Worksheet 对象的 Range 属性，将返回一个 Range 对象，它代表一个单元格或单元格区域。
- Cells 属性，使用 Worksheet 对象的 Cells 属性，将返回一个 Range 对象，它代表工作表中的所有单元格（不仅仅是当前使用的单元格）。
- Offset 属性，使用已有 Range 对象的 Offset 属性，将返回一个新的 Range 对象，它代表位于指定单元格区域的一定的偏移量位置上的区域。
- Union 方法，使用 Application 对象的 Union 方法，将返回两个或多个区域的合并区域。

10.4.3 隐含使用对象

前面介绍了对象层次结构的概念，通过这个层次结构就可以从对象的顶层一直追溯到要操作的对象。例如，要将单元格 A1 的值设置为“对象引用层次结构”，可使用以下

代码：

```
Application.Workbooks(1).Worksheets(1).Range("A1").Value = "对象引用层次结构"
```

以上代码使用了完整的层次路径来引用单元格对象 A1，这种方式在语法上是完全正确的，但是如果没有特殊需要，一般不需要这样编写代码。

其实，在引用 Excel 对象时，应该从系统能够确定与所需对象的层次最相近的对象开始引用。因为 Application 对象代表了正在运行的 Excel 本身。因此，如果当前代码是在 Excel 中执行，就可以隐含 Application 对象的引用，可将上面的代码改为以下形式：

```
Workbooks(1).Worksheets(1).Range("A1").Value = "对象引用层次结构"
```

如果用户要修改的单元格的位置是在活动工作簿中，则可将代码改为如下形式：

```
ActiveWorkbook.Worksheets(1).Range("A1").Value = "对象引用层次结构"
```

其中 ActiveWorkbook 为 Application 对象的一个属性，用来返回当前活动工作簿的引用。

同样，如果要对当前工作表进行修改，可使用以下代码：

```
ActiveWorksheet.Range("A1").Value = "对象引用层次结构"
```

如果代码是在工作表 Worksheets(1)中编写的，还可使用以下形式：

```
Range("A1").Value = "对象引用层次结构"
```

在 Excel 中，还可以使用 Selection 来引用活动窗口中选定的对象。因此，如果要设置当前选中单元格 A1 的值，可使用以下代码：

```
Selection.Value = "对象引用层次结构"
```

10.5　使用对象浏览器

Excel 的对象模型中有二百多个对象，每个对象又具有数量不等的属性、方法、事件和常数，对于开发人员来说，要全部记住这些内容是非常困难的。VBE 中提供了【对象浏览器】对话框，可帮助用户查看对象的属性、方法、事件及常数变量等内容。

本节介绍【对象浏览器】的使用方法。

10.5.1　认识对象浏览器

通过【对象浏览器】可浏览工程中所有可获得的对象并查看它们的属性、方法以及事件。此外还可查看工程中可从对象库获得的过程以及常数。也可用【对象浏览器】去搜索和使用用户自己所创建的对象，其他应用程序的对象也可用它来浏览。

可以按以下步骤使用【对象浏览器】中的功能：

（1）在 VBE 中激活一个模块。

（2）单击主菜单【查看】|【对象浏览器】命令（或按快捷键 F2），打开【对象浏览器】窗口，如图 10-4 所示。

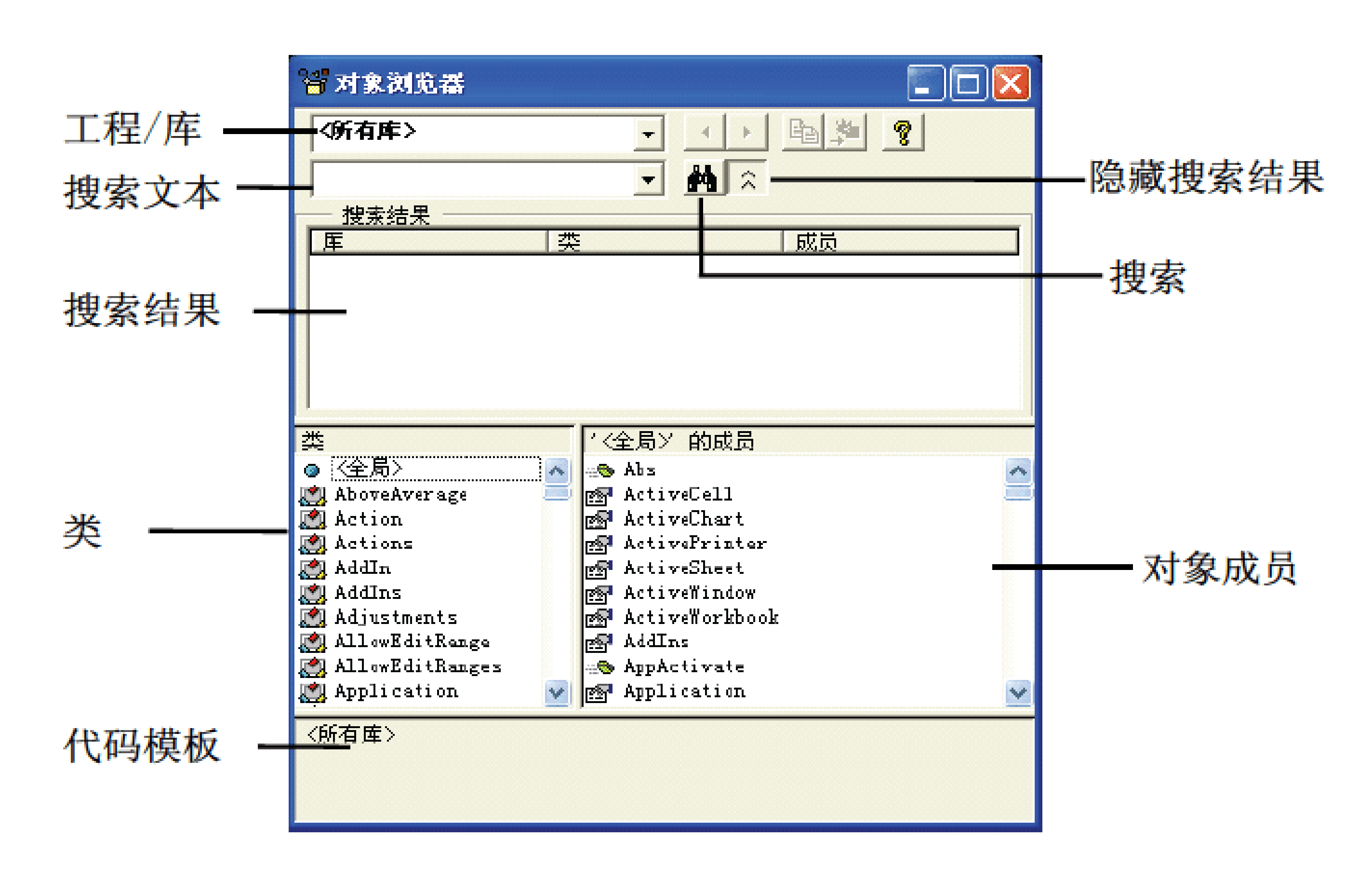


图 10-4 【对象浏览器】窗口

（3）在【工程/库】列表中选定所要查看的工程或程序库名称。

（4）使用【类】列表选定类；使用【成员】列表选定类或工程中的特定成员。

（5）查看在窗口底端的详细资料区选定的工程或类的信息。

（6）使用【帮助】按钮显示所选类或成员的帮助主题。

从图中看出，对象浏览器分为三个主要部分：顶部为搜索部分、中间为搜索结果部分、底部为选中对象成员的代码模板。

下面简单介绍各部件的功能。

- 【工程/库】下拉列表框：列出了所有可用于当前 VBA 工程的所有库和工程的名称。【库】是包含应用程序里相关对象信息的专门文件。新的库可以通过【引用】对话框（单击主菜单【工具】|【引用】命令）来添加。“<所有库>”列出了已安装在计算机中所有库的对象。当选择 Excel 库时，仅仅能看到在 Excel 里执行的对象名称。VBA 库列出了所有能在 VBA 里执行的对象名称。
- 【搜索文本】文本框：在【工程/库】下拉列表框的下方，有一个【搜索文本】文本框，可以快速地在指定库里查找信息。这个地方会记住最近搜索的项目，直到关闭此工程为止。在该文本框中输入字符串时，可以使用标准的 VB 通配符。如果要查找完全相符的字符串，可以在对象浏览器的任何地方右击，从快捷菜单中选择【全字匹配】命令，如图 10-5 所示。
- 【向后】按钮：可以向后回到前一个类及成员列表。每单击一次便向后一个选项，直到最后。
- 【向前】按钮：每次单击可以重复原本选择的类及成员列表，直到选择列表用完。

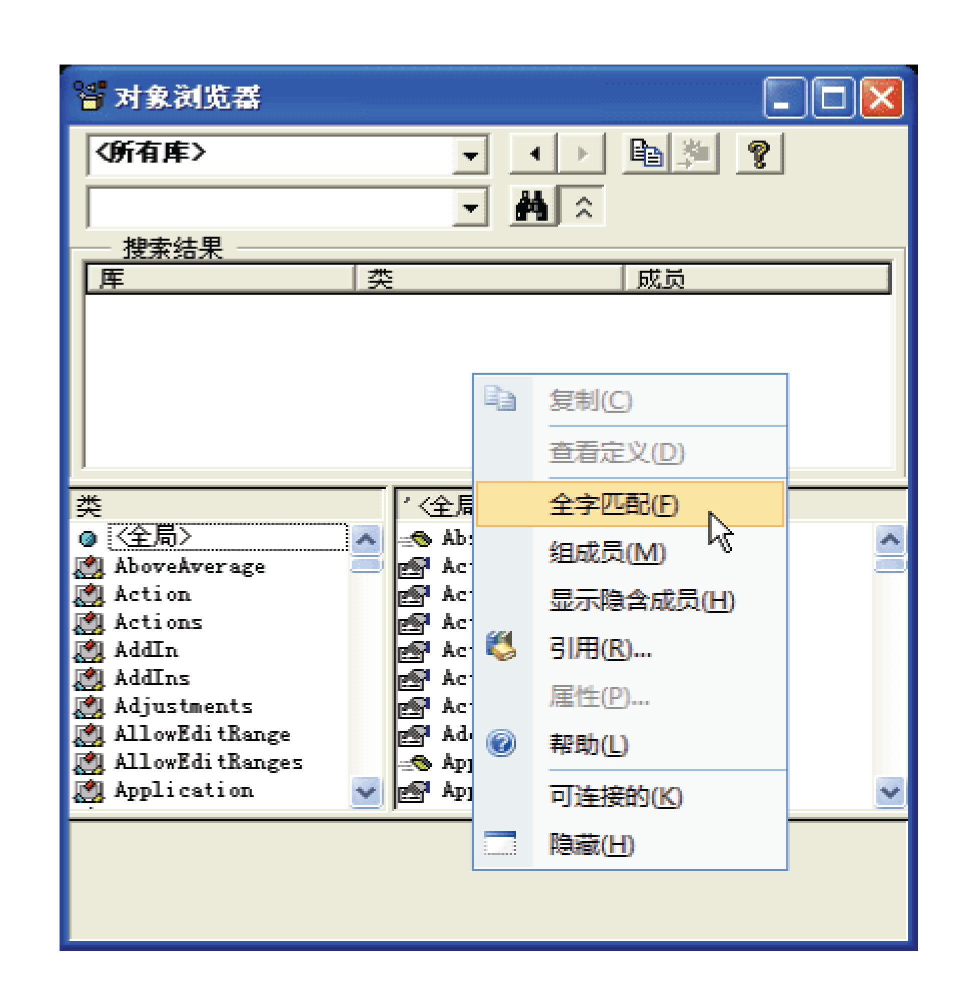


图 10-5 【全字匹配】命令

- 【复制到剪贴板】按钮：将成员列表中的选择或详细框中的文本复制到剪贴板。可在之后将选择贴到代码中。
- 【查看定义】按钮：将光标移到【代码】窗口中，定义成员列表或类列表中选定的位置。
- 【帮助】按钮：显示在类或成员列表中选定工程的联机帮助主题。
- 【搜索】按钮：在选定库中搜索在【搜索文本】框中键入的字符串，并且打开有适当信息列表的【搜索结果】列表框，在其中显示符合搜索文本框中所输入的搜索条件的库、类和成员。
- 【显示/隐藏搜索结果】按钮：打开或隐藏【搜索结果】列表框。【搜索结果】列表框改变为显示从【工程/库】列表中所选出的工程或库的搜索结果。搜索结果会默认的按类型创建组并从 A 到 Z 排列。
- 【搜索结果列表】：显示搜索字符串所包含工程的对应库、类及成员。【搜索结果】列表框在改变【工程/库】下拉列表框中的选择时改变。
- 【类】列表框：显示在【工程/库】下拉列表框中选定的库或工程中所有可用的类。如果有代码编写的类，则该类会以粗体方式显示。这个列表的开头都是“<globals>”，是可以全局访问的成员列表。如果选择了类但没有选择特定的成员，则会得到默认成员。默认的成员以星号（*）或是以此成员特定的默认图标作为标识。
- 【成员列表】：显示在【类】列表框中所选类的元素属性、事件、方法和常数。用代码编写的方法、属性、事件或常数会以粗体显示。默认情况下，将属性、事件、方法和常数按字母顺序排列，在如图 10-6 所示的快捷菜单中，选择【组成员】命令后，将改变显示顺序（以属性、方法、事件和常数分组，每组再按字母顺序排列）。

- 【代码模板】：对象浏览器窗口底部显示所选成员定义的代码模板。如果单击代码模板里绿色链接文本，就可以在对象浏览器窗口中快速跳到所选成员的类或库。代码模板里的文本可以复制到 Windows 剪切板并且粘贴到代码窗口中。如果对象浏览器打开时代码窗口是可见的，那么只需选中代码模板里的文本，直接拖曳到代码窗口就行了。

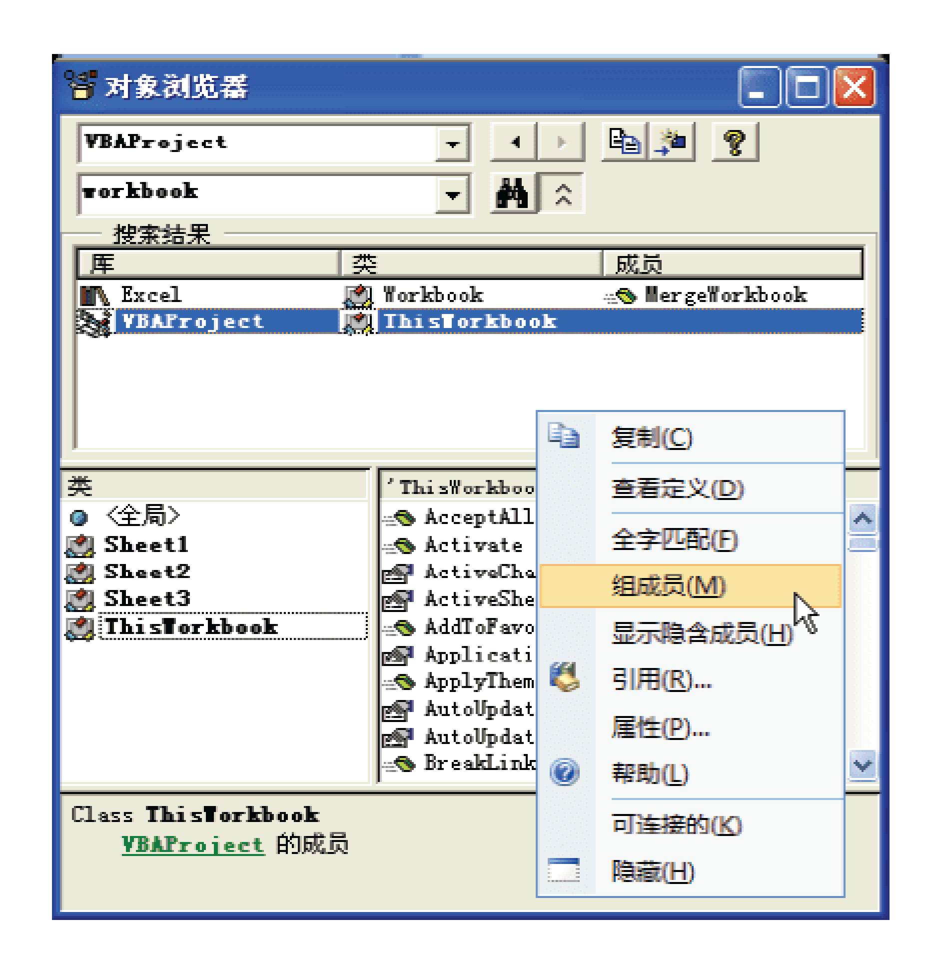


图 10-6　按【组成员】命令排列

10.5.2　用对象浏览器查看对象成员

了解了【对象浏览器】对话框及各部件的作用后，就可使用该对话框来查看对象成员。

例如，Range 对象是 VBA 中最常用的一个对象，下面通过【对象浏览器】窗口来熟悉 Range 对象的属性、方法、事件和常数。

（1）启动 Excel，并按快捷键 Alt+F11 进入 VBE。

（2）按 F2 键打开【对象浏览器】窗口。

（3）在【工程/库】下拉列表框中选择 Excel。

（4）在【搜索文本】框中输入 Range 关键字，并单击右侧的【搜索】按钮，得到如图 10-7 所示的结果。在【搜索结果】列表框中显示了与关键字 Range 完全匹配的选项，如 Range 类，以及许多类的 Range 成员。

（5）在【搜索结果】列表框中单击第一个选项（Range 类），下方将显示出 Range 的成员，如图 10-7 所示。

提示： 成员列表框中以不同的图标表示属性、方法、事件或常数。图标为属性，图标为方法，图标为事件，图标为常数。

（6）在成员列表中单击方法 AutoFit，在下方的【代码模板】中将显示该方法的定义原型，如图 10-8 所示。

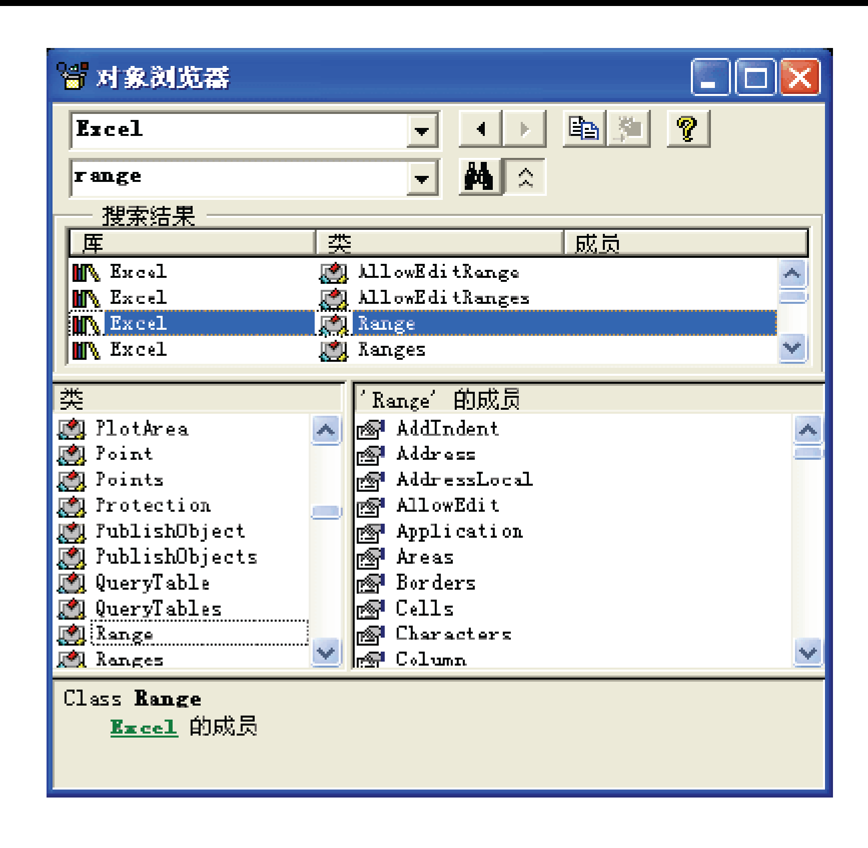


图 10-7　搜索 Range 对象

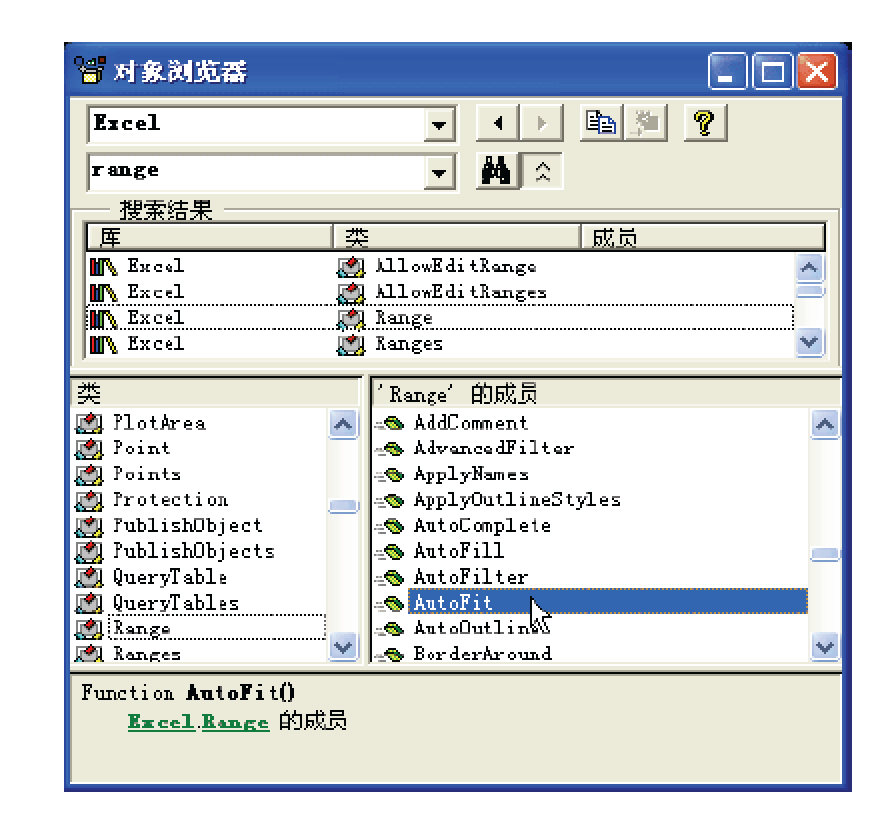


图 10-8　查看成员

（7）单击右上角的【帮助】按钮，可打开【Excel 帮助】窗口，并显示当前选中成员的帮助信息，如图 10-9 所示。



图 10-9　显示帮助信息

（8）在【对象浏览器】窗口中单击上方的【复制到剪贴板】按钮，将代码复制到剪贴板中。

（9）在【代码】窗口中按快捷键 Ctrl+V 即可将其粘贴到当前位置，如图 10-10 所示。

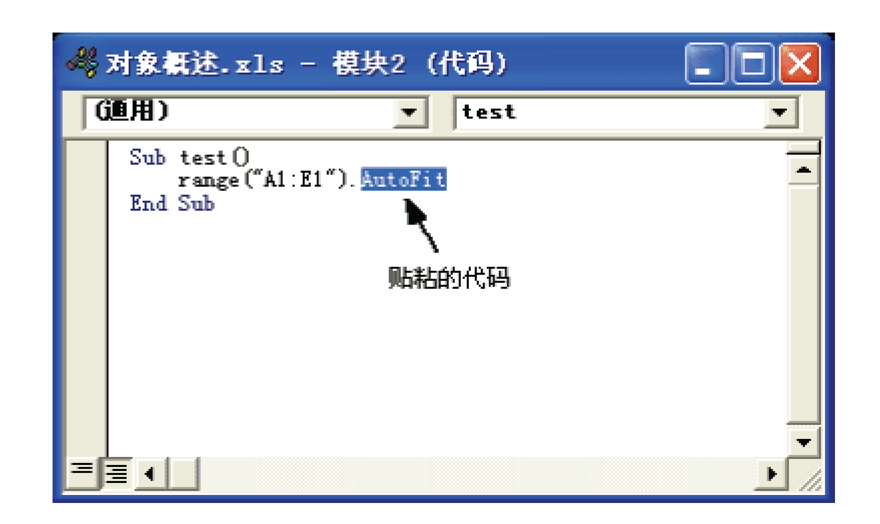


图 10-10　粘贴代码

第 11 章　使用 Application 对象

Application 对象是 Excel 对象模型的最顶层对象，代表 Excel 应用程序本身。Application 对象提供了大量的属性、方法和事件，供用户操作控制 Excel 程序。

11.1　了解 Application 对象

通过 Application 对象提供的属性、方法和事件可控制 Excel 应用程序的外观状态，响应用户的操作。本节的前面内容将简单介绍 Application 对象的常用属性和方法，后面将以实例演示控制 Application 对象的代码。

11.1.1　Application 对象常用属性

Application 对象提供了很多属性，通过这些属性可完成控制 Excel 的状态、获取对象的引用等操作。

1．控制Excel状态

控制 Excel 状态的常用属性有以下几个。

- DisplayFormulaBar 属性：如果该属性值为 True，则显示编辑栏。
- DisplayScrollBars 属性：如果该属性值为 True，则滚动条在所有工作簿中显示。
- DisplayStatusBar 属性：如果该属性值为 True，则显示状态栏。
- ScreenUpdating 属性：如果该属性值为 True，则启用屏幕更新。
- StatusBar 属性：返回或设置状态栏中的文字。
- Visible 属性：返回或设置一个 Boolean 值，其确定对象是否可见。
- WindowState 属性：返回或设置窗口的状态。

2．获取对象的引用

许多 Application 对象的属性可用来返回其他的对象。返回对象引用的属性主要有以下几种。

- ActiveCell 属性：返回一个 Range 对象，它代表活动窗口（最上方的窗口）或指定窗口中的活动单元格。如果窗口中没有显示工作表，则此属性无效。
- ActiveChart 属性：返回一个 Chart 对象，它代表活动图表（嵌入式图表或图表工作

表）。嵌入式图表在被选中或激活时被认为是活动的。当没有图表处于活动状态时，此属性返回 Nothing。

- ActiveSheet 属性：返回一个对象，它代表活动工作簿中或指定的窗口指定的工作簿中的活动工作表（最上面的工作表）。如果没有活动的工作表，则返回 Nothing。
- ActiveWindow 属性：返回一个 Window 对象，该对象表示活动窗口（顶部窗口）。如果没有打开的窗口，则返回 Nothing。
- ActiveWorkbook 属性：返回一个 Workbook 对象，该对象表示活动窗口（顶部窗口）中的工作簿。如果没有打开的窗口，或者“信息”、“剪贴板”的窗口为活动窗口，则返回 Nothing。
- Cells 属性：返回一个 Range 对象，它代表活动工作表中的所有列。因为 Item 属性是 Range 对象的默认属性，所以可以在 Cells 关键字后面紧接着指定行和列索引。
- Selection 属性：为 Application 对象返回在活动窗口中选定的对象。返回的对象类型取决于当前所选内容（例如，如果选择了单元格，此属性将返回 Range 对象）。如果未选择任何内容，Selection 属性将返回 Nothing。
- Sheets 属性：返回一个 Sheets 集合，它代表活动工作簿中所有的工作表。
- Workbooks 属性：返回一个 Workbooks 集合，该集合表示所有打开的工作簿。
- WorksheetFunction 属性：用作可以从 VB 中调用的 Excel 工作表函数的容器。

11.1.2　Application 对象常用方法

Application 对象提供了许多允许执行操作的方法，例如，重新计算当前数据、撤销操作等。下面列出常用的方法。

- Calculate 方法：计算所有打开的工作簿、工作簿中的某个特定工作表或工作表指定区域中的单元格。
- Evaluate 方法：将一个 Excel 名称转换为一个对象或者一个值。该方法允许以字符串的形式创建引用，并且在需要时将其转换成一个实际对象引用（或求出表达式的值）。
- Quit 方法：退出 Excel 程序。使用此方法时，如果未保存的工作簿处于打开状态，则 Excel 将显示一个对话框，询问是否要保存所作更改。要防止发生这种情况，可以在使用 Quit 方法前保存所有工作簿或将 DisplayAlerts 属性设置为 False。如果该属性为 False，则在 Excel 退出时，即使有未保存的工作簿，也不会显示对话框，而且不会不保存就退出。
- OnTime 方法：安排一个过程在将来的特定时间运行（既可以是具体指定的某个时间，也可以是指定的一段时间之后）。
- Undo 方法：撤销最后一次用户界面操作。本方法仅撤销运行该宏之前的最后一个用户操作，并且必须将其放到宏的第一行，不能撤销 VBA 命令。
- Union 方法：返回两个或多个区域的合并区域。

11.1.3 Application 对象常用事件

在 Excel 的对象模型中，大部分对象都提供了事件接口。在大部分情况下，通过 Workbook、Worksheet、Range 等对象提供的事件接口能满足程序设计的需要，用户可以不使用 Application 对象的事件过程。

大部分 Application 对象的事件与 Workbook、Worksheet 等对象的事件相似，常用的事件有下面几种。

- ❑ NewWorkbook 事件：当新建一个工作簿时发生此事件。
- ❑ SheetActivate 事件：当激活任何工作表时发生此事件。
- ❑ SheetBeforeDoubleClick 事件，当双击任何工作表时发生此事件，此事件先于默认的双击操作发生。
- ❑ SheetChange 事件：当用户或外部链接更改了任何工作表中的单元格时发生此事件。
- ❑ SheetDeactivate 事件：当任何工作表被停用时发生此事件。
- ❑ WindowActivate 事件：工作簿窗口被激活时，将发生此事件。
- ❑ WindowDeactivate 事件：任何工作簿窗口被停用时将发生此事件。
- ❑ WindowResize 事件：任何工作簿窗口调整大小时将发生此事件。
- ❑ WorkbookActivate 事件：当激活任何一个工作簿时发生此事件。
- ❑ WorkbookBeforeClose 事件：当任何一个打开的工作簿关闭之前立即发生此事件。
- ❑ WorkbookBeforeSave 事件：在保存任何一个打开的工作簿之前发生此事件。
- ❑ WorkbookNewSheet 事件：在任何打开的工作簿中新建工作表时发生此事件。
- ❑ WorkbookOpen 事件：当打开一个工作簿时发生此事件。

对于 Application 对象的事件：还需要通过特殊的设置方法才能将其激活。在本章后面的内容中将会详细介绍。

11.2 设置应用程序选项

Application 对象代表整个 Excel 应用程序，通过 Application 对象的属性就可设置应用程序的各个选项。

11.2.1 设置主窗口标题栏

正常情况下，Excel 应用程序的标题栏中将显示 Microsoft Excel。如果在 Excel 中设计了一个应用程序，可能希望标题栏显示的内容为应用程序的名称，例如“工资管理系统”。

使用 Application 对象的 Caption 属性，可以改变 Excel 主窗口标题栏中显示的名称。一般将这类代码放在工作簿的 Open 事件中，打开工作簿即可将其设置需要的名称。例如，以下代码将标题栏设置为“工资管理系统”。

```
Private Sub Workbook_Open()
    Application.Caption = "工资管理系统"
End Sub
```

关闭 Excel，再重新打开包含以上代码的 Excel 工作簿（或在 VBE 环境中执行以上代码），可得到如图 11-1 所示的标题栏名称。



图 11-1　设置标题栏

Caption 属性是针对整个 Excel 应用程序的设置，退出包含以上代码的工作簿后，Excel 的标题栏仍将显示修改后的标题名称。因此，一般在工作簿的 BeforeClose 事件中编写以下代码，来恢复 Excel 应用程序的标题栏信息。

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    Application.Caption = ""
End Sub
```

提示： 将 Caption 属性设置为空，则可使 Caption 属性返回 Microsoft Excel。

11.2.2　控制状态栏

通过 StatusBar 属性可返回或设置状态栏中的文字。一般在应用程序需要进行大量运算的工作时，可以在状态栏中显示相应的提示信息。否则，应用程序长时间无响应，用户会以为计算机系统死机。

例如，下面的代码在状态栏动态显示正在处理的数据行数，最后设置 StatusBar 属性值为 False，可将状态栏中的内容清除。

```
Sub 控制状态栏()
    Dim i As Long
    For i = 1 To ActiveSheet.Rows.Count
        If i Mod 100 = 0 Then
            Application.StatusBar = "正在处理第 " & i & " 行的数据，请稍候！"
        End If
    Next
    Application.StatusBar = False
End Sub
```

执行以上代码，状态栏的显示如图 11-2 所示。

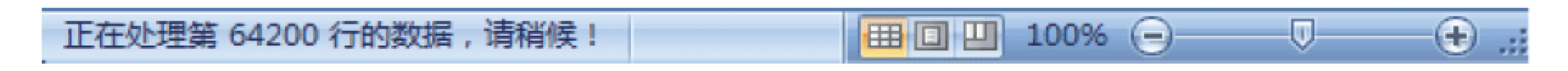


图 11-2　控制状态栏

除了设置 StatusBar 的属性为 False 来清除状态栏中的数据外，还可使用 DisplayStatusBar 属性。例如，可使用以下语句清除状态栏中的内容。

```
Application.DisplayStatusBar = oldStatusBar
```

11.2.3 控制编辑栏

在如图 11-3 所示的【Excel 选项】对话框中，可选中或取消【显示编辑栏】前面的复选框来控制是否显示编辑栏。

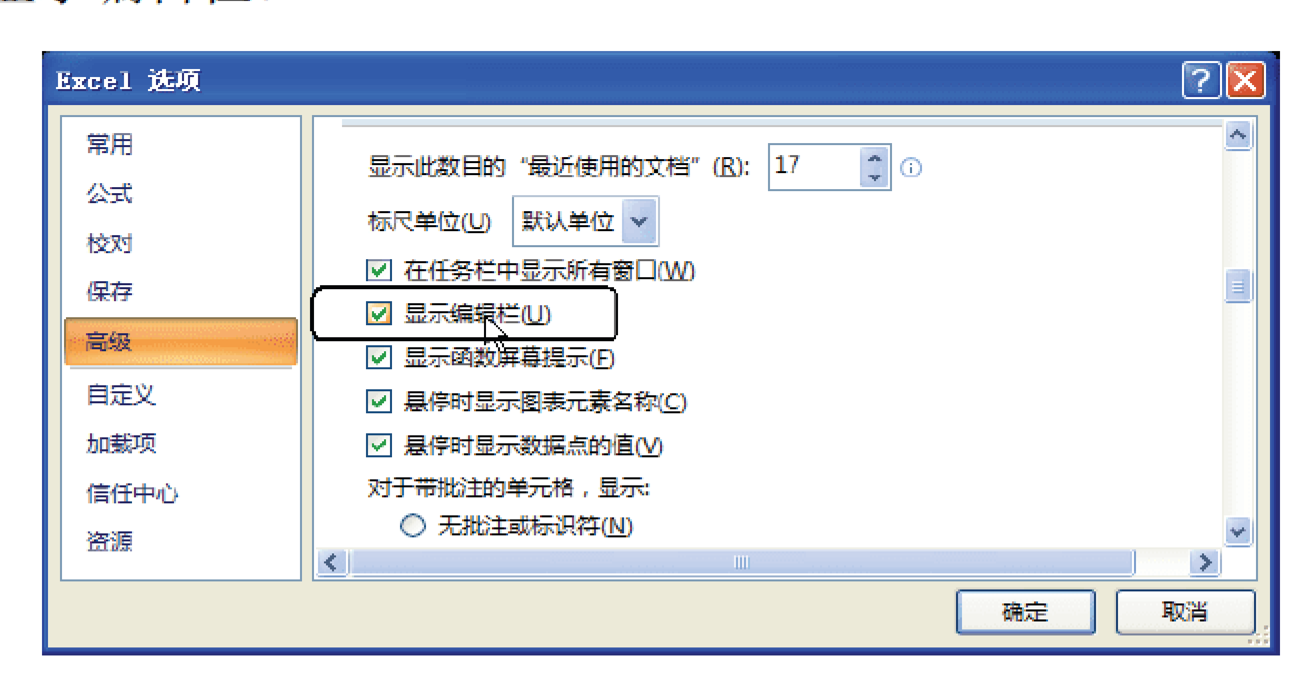


图 11-3 【Excel 选项】对话框

通过 VBA 编写代码，也可通过 DisplayFormulaBar 属性来控制【编辑栏】的显示状态。如果该属性值为 True，则显示编辑栏。

使用下面的代码，可控制编辑栏的显示状态。

```
Sub 控制编辑栏()
    With Application
        If .DisplayFormulaBar Then
            .DisplayFormulaBar = False
        Else
            .DisplayFormulaBar = True
        End If
    End With
End Sub
```

以上代码首先通过 DisplayFormulaBar 属性获取当前编辑栏的状态，再决定是显示还是隐藏编辑栏。

提示： 用户可以在 Excel 工作表中添加一个按钮，用来执行上面的宏，以方便地控制编辑栏的显示状态。

11.2.4 控制鼠标指针形状

在 Windows 操作系统中，可以通过鼠标指针的形状获得系统当前的状态。在 Excel 工

作簿中，鼠标指针的形状有 4 种样式，在 VBA 程序中可以分别设置为以下 4 个常量之一。

- xlDefault：默认指针，值为–4143。
- xlNorthwestArrow：西北向箭头指针，值为 1。
- xlWait：沙漏型指针，值为 2。
- xlIBeam：I 形指针，值为 3。

通过 Application 对象的 Cursor 属性可获取或设置 Excel 中鼠标指针的外观。

注意：当 VBA 代码停止运行时，Cursor 属性不会自动重设。在 VBA 代码停止运行前，应将指针重设为 xlDefault。

例如，运行以下的代码，将弹出对话框提示显示第几种鼠标形状，每显示一种指针形状暂停 5 秒钟，再接着显示下一种指针形状，最后恢复默认指针形状。

```
Sub 显示鼠标指针形状()
    Dim i As Integer
    For i = 1 To 3
        MsgBox "显示第 " & i & " 种鼠标指针形状！", vbInformation + vbOKOnly
        Application.Cursor = i
        st = Timer
        Do While Timer <= st + 5
            DoEvents
        Loop
    Next
    MsgBox "恢复默认鼠标指针形状！", vbInformation + vbOKOnly
    Application.Cursor = xlDefault
End Sub
```

11.3　控制应用程序

通过 Application 对象的相关属性，还可对 Excel 应用程序的运行状态进行控制，例如，关闭屏幕刷新、禁止弹出警告对话框、重新计算工作簿等。

11.3.1　控制屏幕更新

在默认情况下，Excel 每执行一个操作就会更新一次屏幕，以显示出执行的结果。在使用 VBA 操作时，每执行一次操作 Excel 也将更新一次屏幕，这样将导致程序运行速度下降。关闭屏幕更新可以加快程序的执行速度，这样将看不到程序的执行过程，但程序的执行速度加快了。

将 Application 对象的 ScreenUpdating 属性设置为 True，将启用屏幕更新，设置为 False 时，将关闭屏幕更新。

⚠注意：当代码结束运行后，应将 ScreenUpdating 属性设置为 True。

为了演示关闭屏幕更新后 VBA 代码执行速度的提高程度，需要一个大的循环才能体现出来。

例如，以下代码将工作表 Sheet1 中的偶数行隐藏。为了对比提速的时间，将程序执行时间保存在数组中，第一次是屏幕更新为打开状态时操作所用的时间；第二次执行是屏幕更新为关闭状态时操作所用的时间。

```
Sub 屏幕更新()
    Dim aTime(2)
    Application.ScreenUpdating = True
    For i = 1 To 2
        If i = 2 Then Application.ScreenUpdating = False
        Worksheets(i).Activate
        startTime = Timer
        For j = 1 To ActiveSheet.Rows.Count
            If j Mod 2 = 0 Then
                Rows(j).Hidden = True
            End If
        Next j
        stopTime = Timer
        aTime(i) = stopTime - startTime
    Next i
    Application.ScreenUpdating = True
    MsgBox "打开屏幕更新,程序执行的时间: " & aTime(1) & " 秒" & Chr(13) & _
        "关闭屏幕更新,程序执行的时间: " & aTime(2) & " 秒"
End Sub
```

程序的执行结果如图 11-4 所示。由程序结果可以看出，将 ScreenUpdating 属性设置为 True 时，程序执行的时间为 128 秒，而将 ScreenUpdating 属性设置为 False 时，程序执行的时间约为 9 秒。由此可以看出程序执行速度的差别为 10 多倍。

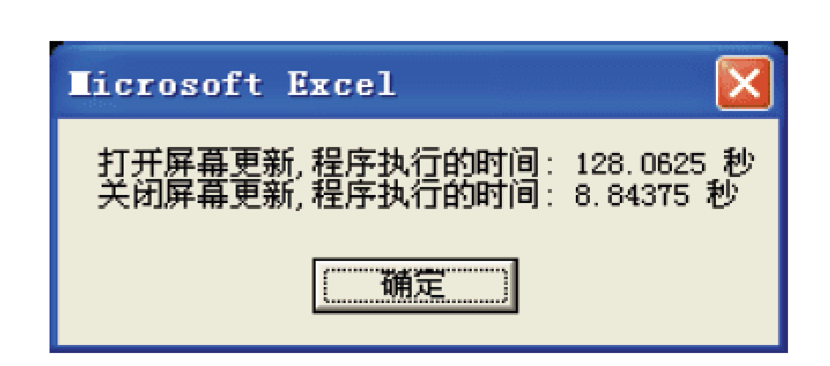


图 11-4　关闭/打开屏幕更新对程序的影响

⚠提示：如果在 VBE 中执行以上代码（Excel 界面隐藏在后面），关闭或打开屏幕更新对程序的影响不大，因为这时 Excel 窗口处于隐藏状态。

11.3.2　控制报警信息

在正常情况下，当用户删除工作簿中的工作表时，将弹出如图 11-5 所示的对话框，提

示用户删除工作表后数据将不能恢复。单击【删除】按钮后可删除工作表。



图 11-5　删除提示

在交付给用户的 Excel 应用程序中，一般不希望弹出上图所示的警告信息对话框。这时，可通过设置 Application 对象的 DisplayAlerts 属性，来控制 Excel 是否显示 VBA 程序执行过程中的警告和消息。

该属性的默认值为 True。如果不想在程序运行过程中被无穷无尽的提示和警告消息所困扰，应将本属性设置为 False，这样每次出现需用户应答的消息时，Excel 将选择默认应答。

如果使用工作簿的 SaveAs 方法覆盖现有文件，"覆盖"警告默认为 No，当 DisplayAlerts 属性设置等于 False 时，Excel 选择 Yes 响应。

提示：如果将该属性设置为 False，则在代码运行结束后，Excel 将该属性设置为 True，除非正在运行交叉处理代码。

例如，以下代码将删除活动工作表，并不弹出警告信息对话框。

```
Sub 删除工作表()
    Application.DisplayAlerts = False
    ActiveSheet.Delete
    Application.DisplayAlerts = True
End Sub
```

以上代码首先设置 DisplayAlerts 属性值为 False，再执行工作表的删除方法，最后将 DisplayAlerts 属性设置为 True，结束程序。

11.3.3　显示最近使用的文档

单击 Excel 2007 的【Office 按钮】，将打开下拉菜单。在该下拉菜单右侧显示了最近使用的文档，如图 11-6 所示。

通过 Application 对象的 RecentFiles 属性，可以返回一个 RecentFiles 集合，该集合表示最近使用的文件列表。通过该集合的属性可设置最近使用的文档列表，常用的属性有以下两种。

- Maximum 属性：返回或设置最近使用文件清单中文件数目的上限。可为 0～9 之间的数字。
- Count 属性：集合中对象（最近使用的文件）的数量。



图 11-6　最近使用的文档

例如，以下代码将最近使用文档名称填入工作表的第 1 列中。

```
Sub 最近使用文档()
    Dim i As Long, j As Long
    Dim r As RecentFile
    ActiveSheet.Columns(1).Clear

    i = 1
    For Each r In Application.RecentFiles
       ActiveSheet.Cells(i, 1) = r.Name
       i = i + 1
    Next
End Sub
```

11.3.4　模拟键盘输入

使用 Application 对象的 SendKeys 方法，可将击键发送给活动应用程序。该方法的语法格式如下：

```
Application.SendKeys(Keys, Wait)
```

各参数的含义分别如下。

- Keys：要以文本形式发送给应用程序的键或组合键。
- Wait：如果为 True，则 Excel 会等到处理完按键后将控件返回给宏；如果为 False（或者省略该参数），则继续运行宏而不等到处理完按键。

参数 Keys 可指定任何单个键或与 Alt、Ctrl、Shift 的组合键（或者这些键的组合）。每个键可用一个或多个字符表示。例如，"a"表示字符 a，或者"{ENTER}"表示 Enter。

若要指定那些没有屏幕回显该字符的键（例如，Enter 或 Tab），则需要使用特殊的代码来表示相应的键。具体值可参见帮助系统。

下面的代码首先打开【记事本】窗口，并使用 AppActivate 语句将【记事本】程序激

活，接着使用 SendKeys 发送字符给【记事本】程序。

```
Sub 模拟输入()
    Dim dReturnValue As Double
    dReturnValue = Shell("NOTEPAD.EXE", 1)       '打开记事本
    AppActivate dReturnValue                     '激活应用程序
    Application.SendKeys "~", True
    Application.SendKeys "Keybord input demo :", True
    Application.SendKeys "~", True
    Application.SendKeys " Excel 2007 VBA ! ", True
End Sub
```

提示：在 SendKeys 方法中，“~”表示 Enter 键。

运行以上代码，将打开【记事本】窗口，并显示如图 11-7 所示的效果。

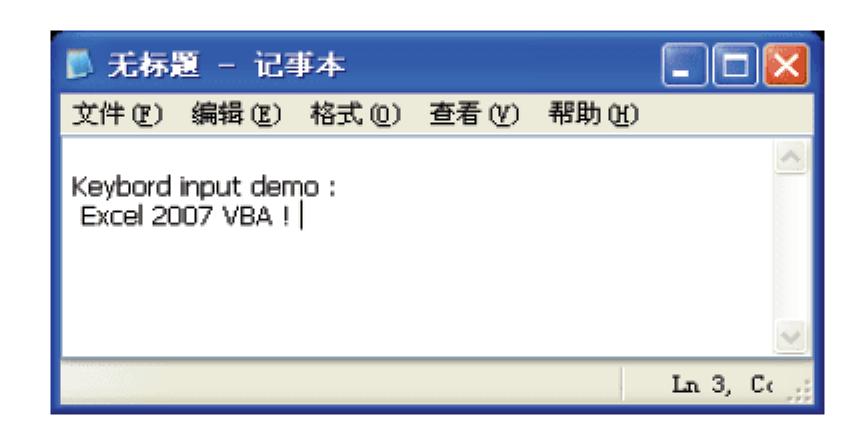


图 11-7 【记事本】窗口

注意：以上代码需要 Excel 界面中执行。若在 VBE 环境中执行，则 SendKeys 方法将字符发送到 VBE 的【代码】窗口中。

11.3.5　定时执行过程

使用 Application 对象的 OnTime 方法，可安排一个过程在将来的特定时间运行（既可以是具体指定的某个时间，也可以是指定的一段时间之后）。例如，设置 20 秒后运用过程 Test1 可使用以下代码：

```
Application.OnTime Now + TimeValue("00:00:20"), "Test1"
```

早上 8 点运行过程 Test2，代码为：

```
Application.OnTime TimeValue("08:00:00"), "Test2"
```

要撤销运行 OnTime 设置的过程，需要将 Schedule 参数设置为 Flase。如撤销前一个表达式对 OnTime 的设置，则代码如下：

```
Application.OnTime   EarliestTime:=TimeValue("08:00:00"),   Procedure:=
"Test2", Schedule:=False
```

以下代码将进行整点报时：

```
Sub starttime()
    Application.OnTime EarliestTime:=TimeSerial((Hour(Now) + 1) Mod 24, 0,
    0), _
        Procedure:="starttime"
    MsgBox "现在时间是: " & Hour(Now) & " 点! "
End Sub
```

以上代码首先通过 OnTime 方法设置整点时（小时数与 24 进行模运算）调用的过程，接着显示一个对话框进行报时。

使用以下代码取消整点报时功能：

```
Sub endtime()
    On Error Resume Next
    Application.OnTime EarliestTime:=TimeSerial((Hour(Now) + 1) Mod 24, 0,
    0), _
        Procedure:="starttime", schedule:=False
End Sub
```

在执行以上代码时，如果没有使用 OnTime 设置调用对应的过程 starttime，则程序将出错，所以第 1 条语句设置错误捕捉。

11.3.6 自定义功能键

使用 Application 对象的 OnKey 方法，可在设定的特定键或组合键被按下时，运行指定的过程。其语法格式如下：

```
Application.OnKey(Key, Procedure)
```

各参数的含义分别如下。

- Key：表示要按的按键的字符串，其表示方法与本章的 11.3.4 节中的 SendKeys 方法相同。
- Procedure：表示要运行的过程名称的字符串。如果 Procedure 为空文本（""），则按 Key 时不发生任何操作。如果省略 Procedure 参数，则 Key 恢复为 Excel 中的正常结果，同时清除先前使用 OnKey 方法所做的特殊击键设置。

下面的代码自定义组合键 Alt+.（字符“.”的上档键为符号“>”）可使工作表向下翻页，组合键 Alt+,（字符“,”的上档键为符号“<”）可使工作表向上翻页。

首先编写“设置自定义功能键”过程的代码如下：

```
Sub 设置自定义功能键()
    Application.OnKey "%.", "NextPage"
    Application.OnKey "%,", "PrePage"
End Sub
```

以上代码为两个组合键设置了调用的过程，这两个过程的代码如下：

```
Sub NextPage()
    ActiveWindow.LargeScroll down:=1
End Sub
```

```
Sub PrePage()
    ActiveWindow.LargeScroll up:=1
End Sub
```

当执行“设置自定义功能键”过程后，在 Excel 界面中按组合键 Alt+.即可使工作表向下翻页，组合键 Alt+，可使工作表向上翻页。并且这两个功能键一直有效，要取消自定义组合键的功能，可使用以下代码：

```
Sub 禁止自定义功能键()
    Application.OnKey "%."
    Application.OnKey "%,"
End Sub
```

11.3.7　调用 Excel 工作表函数

使用 Application 对象的 WorksheetFunction 属性，可方便地调用 Excel 工作表函数。例如，以下代码显示单元格区域“A1:A10”的和（使用 Excel 工作表函数 SUM）。

```
Set myRange = Worksheets("Sheet1").Range("A1:C10")
total = Application.WorksheetFunction.Sum(myRange)
```

在程序中也可省略 WorksheetFunction，直接写为 Application.Sum 的形式。

例如，假设在工作表中保存有如图 11-8 所示的股票数据，可在 VBA 中调用 Excel 工作表函数 VLookup 按股票代码查询其价格，具体代码如下：

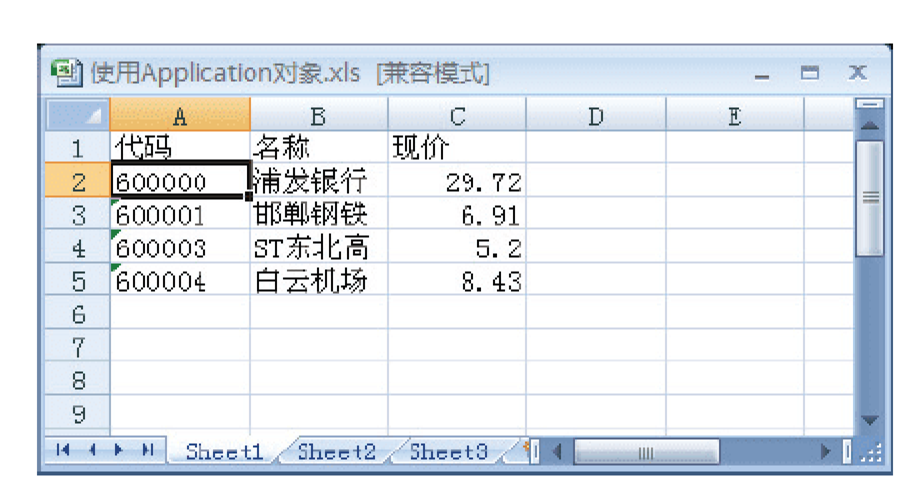


图 11-8　工作表数据

```
Sub 查询股票价格()
    Dim sStock As String, cPrice As Currency
    sStock = InputBox(prompt:="输入股票代码:" & Chr(13) & " (例如:600000) ")
    cPrice = Application.WorksheetFunction.VLookup(sStock, _
        Worksheets("Sheet1").Range("A1:C5"), 3, 0)
    MsgBox "股票" & sStock & "收盘价为: " & cPrice
End Sub
```

使用 WorksheetFunction 对象的 CountIf 方法，可计算区域中满足给定条件的单元格的个数。其语法格式如下：

```
表达式.CountIf(Arg1, Arg2)
```

两个参数的含义分别如下。

- ❑ Arg1：要计算其中满足条件的单元格个数的单元格区域。
- ❑ Arg2：用于定义哪些单元格将被计算在内的条件，其形式可以为数字、表达式、单元格引用或文本。例如，条件可以表示为 32、"32"、">32"、"apples" 或 B4。

提示：可以在条件中使用通配符，包括问号（?）和星号（*）。问号可匹配任意的单个字符；星号可匹配任意一串字符。如果要查找实际的问号或星号，则请在该字符前键入一个波形符（~）。

例如，以下代码使用 CountIf 函数在指定区域生成不重复的随机数：

```
Sub 生成不重复随机数()
    Dim rng As Range, rng1 As Range
    Set rng = Application.InputBox(prompt:="选择要保存不重复随机数的单元格区
    域：", _
        Title:="生成随机数", Type:=8)
    If rng Is Nothing Then Exit Sub
    Randomize
    For Each rng1 In rng                        '选中区域的每个单元格生成随机数
        Do
            rng1 = Int(Rnd * 100 + 1)           '生成 1～100 的随机数
        Loop Until Application.CountIf(rng, rng1) = 1
                                                '循环判断随机数是否有重复
    Next
End Sub
```

以上代码首先让用户选择一个单元格区域，再对每个单元格生成一个随机数。为了生成不重复的随机数，再使用一个循环判断，直到当前单元格的数与所有单元格都不重复为止，然后再生成下一个单元格中的数。

在以上代码中，使用 Application 对象的 InputBox 方法可以显示一个接收用户输入的对话框。与 InputBox 函数不同的是，InputBox 方法可指定输入数据的类型。可以在 Type 参数中设置输入数据的类型，类型值可以为下列值之一或其中几个值的和。

- ❑ 0：公式；
- ❑ 1：数字；
- ❑ 2：文本（字符串）；
- ❑ 4：逻辑值（True 或 False）；
- ❑ 8：单元格引用，作为一个 Range 对象；
- ❑ 16：错误值，例如#N/A；
- ❑ 64：数值数组。

例如，对于一个可接受文本和数字的输入框，将 Type 设置为 1 + 2。

如果 Type 为 8，InputBox 方法将返回一个 Range 对象。必须用 Set 语句将结果指定给一个 Range 对象，例如以下代码：

```
Set myRange = Application.InputBox(prompt := "选择单元格区域", type := 8)
```

如果不使用 Set 语句，此变量将被设置为这个区域的值，而不是 Range 对象本身。

11.3.8　快速跳转

使用 Application 对象的 Goto 方法可选定任意工作簿中的任意区域，并且如果该工作簿未处于活动状态，就激活该工作簿。通过该方法的 Scroll 属性，可以让窗口滚动到目标位置。

例如，以下代码将选择“Sheet2”工作表中的“A1:A10”区域，并将该区域滚动到当前窗口中显示。

```
Sub 快速跳转()
    Application.Goto Reference:=Worksheets("Sheet2").Range("A1:A10"),
    Scroll: =True
End Sub
```

11.3.9　合并单元格区域

使用 Application 对象的 Union 方法，可返回两个或多个区域的合并区域。其语法格式如下：

```
Application.Union(Arg1,Arg2,… ,Arg30)
```

使用该方法时，最少需要 2 个 Range 对象区域作为参数，最多可以合并 30 个 Range 对象区域。例如：

```
Sub 合并区域()
    Worksheets("Sheet3").Activate
    Set unRange = Application.Union(Range("A1:B5"), Range("D1:E5"))
    unRange.Formula = "=RAND()"
End Sub
```

以上代码首先将单元格区域“A1:B5”和“D1:E5”合并为一个 Range 对象，设置该区域对象中各单元格的公式为一个随机函数。执行该部分代码后，将在这 20 个单元格区域（单元格区域“A1:B5”和“D1:E5”共 20 个单元格）中填充随机数。结果如图 11-9 所示。

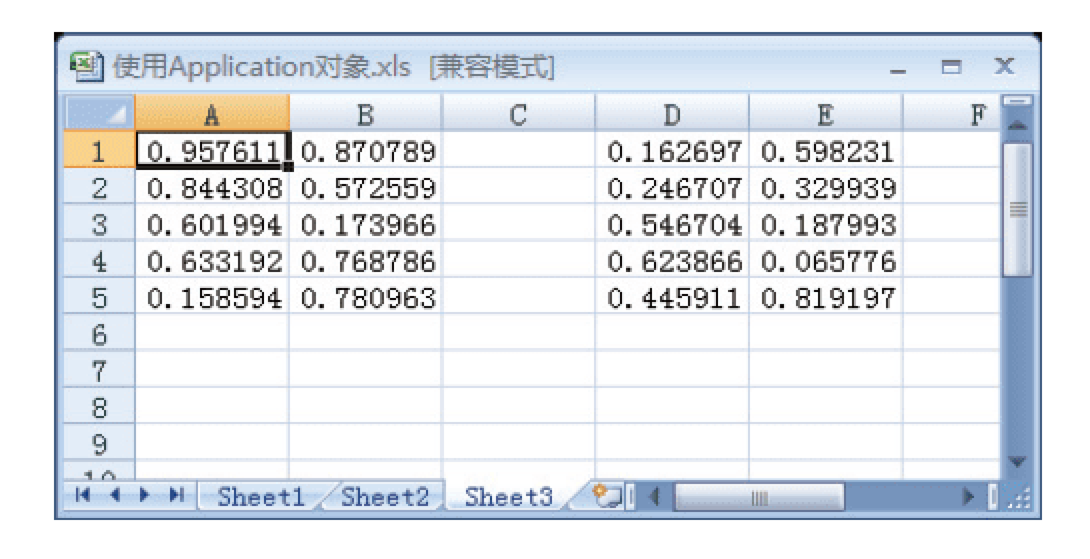


图 11-9　合并区域

11.3.10 激活 Excel 2007 的功能区选项卡

在 Excel 2007 环境中操作时，按一次 Alt 键，在各选项卡下方将显示一个字母，该字母就是激活功能区选项卡的快捷键，如图 11-10 所示。

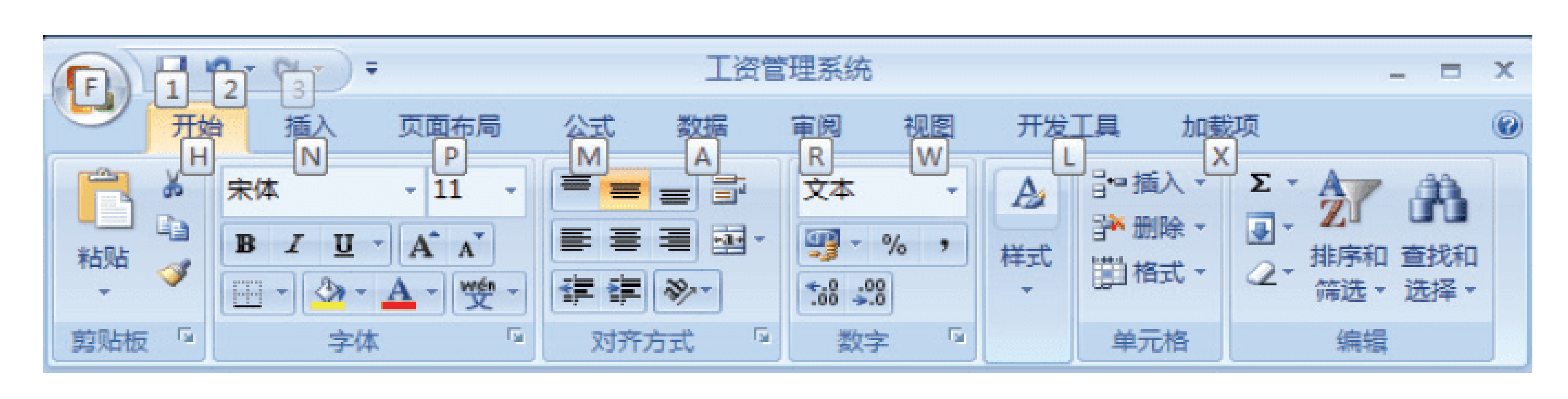


图 11-10 选项卡的快捷键

如图 11-10 所示，按键盘上的组合键 Alt+H 就可激活【开始】选项卡，按组合键 Alt+N 就可激活【插入】选项卡。

在 Excel 2007 中，没有提供从 VBA 中直接激活功能区选项卡的方法。在某些情况下需要通过 VBA 代码选择不同的选项卡，可以使用 SendKeys 方法。

例如，要激活【开始】选项卡，可使用以下语句（模拟键盘按组合键 Alt+H）：

```
Application.SendKeys ("%h")
```

当按下 Alt 键时，将激活各功能的快捷键提示，为了隐藏这些提示，还需再按一次 F6 键，所以应使用以下代码来激活【开始】选项卡：

```
Application.SendKeys ("%h{F6}")
```

同理，要激活【插入】选项卡，可使用以下代码：

```
Application.SendKeys ("%n{F6}")
```

激活其他选项卡的代码与此类似，这里就不再列出。

11.4 处理用户动作

在面向对象的环境中，应用程序响应用户事件即可与用户进行交互。在本章前面简单介绍了 Application 对象的事件。在 Excel 中，要响应 Application 事件，还需要进行特别的设置。本节首先介绍激活 Application 事件的方法，再举例说明 Application 事件的用法。

11.4.1 启用 Application 事件

要激活 Application 事件，需要进行一些特殊的设置，需要使用到类的相关知识，有关类的知识请参考本书第 27 章的介绍，在本节只需按步骤进行设置即可。

启用 Application 事件的具体方法如下：

（1）在 VBE 中单击主菜单【插入】|【类模块】命令，向当前工程中增加一个类模块。

（2）在【属性】窗口中为类模块设置一个名称，该名称无特殊要求，本例设置为 EventClassModule，如图 11-11 所示。



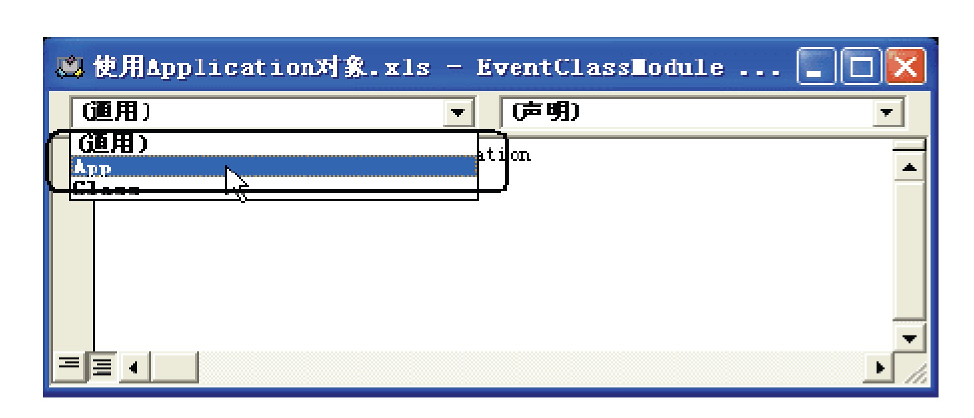
图 11-11　设置类模块名称

（3）确保类模块的【代码】窗口处于打开状态，在类模块【代码】窗口的声明部分编写以下代码，定义一个全局对象变量，用来引用 Application 对象。

```
Public WithEvents App As Application
```

🔔注意：关键字 WithEvents 说明变量 App 是用来响应由 Application 对象触发的事件的对象变量。只有在类模块中才是合法的。使用 WithEvents，可以定义任意个所需的单个变量。

（4）在类模块中声明带有事件的新对象以后，该对象变量将出现在类模块的【对象】下拉列表框中，如图 11-12 左图所示。



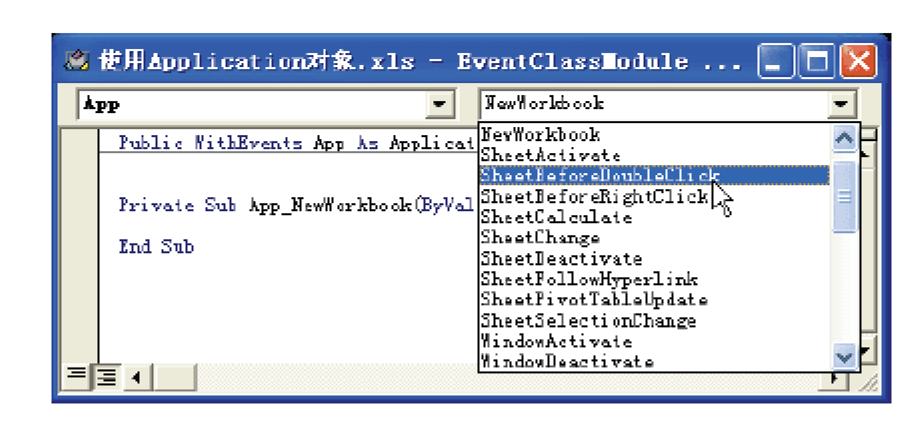


图 11-12　显示对象变量

（5）在【对象】下拉列表框中选择上步声明的变量，该对象的有效事件将列在【过程】下拉列表框中，如图 11-12 右图所示。选择好事件后，将自动插入该事件过程的代码结构，直接在该结构中编写响应事件的代码即可。

（6）经过上面的步骤可为 Application 对象编写响应事件的代码，但此时还不能捕获响应的事件。要捕获事件，必须连接在类模块中声明的对象和 Application 对象。通过单击主菜单【插入】|【模块】命令，向工程中新增一个模块（此处插入的模块名为“模块 2”，根据读者前期的操作不同，该模块的名称可能不同），并在模块的声明部分输入以下代码：

```
Dim X As New EventClassModule
```

🔔注意：EventClassModule 为类模块的名称，若读者的类模块名称不同，这里也需要进行更改。

（7）在“模块 2”中编写以下代码，将类模块中的 App 对象指向 Excel 的 Application 对象，并且类模块中的事件过程在事件发生时将会执行。执行以下代码后，将响应 Application 的事件。

```
Sub 启用Application事件()
    Set X.App = Application
End Sub
```

（8）还可以在“模块 2”中编写以下代码，取消类模块中的 App 对象与 Excel 的 Application 对象的连接，以禁止 Application 事件。

```
Sub 禁止Application事件()
    Set X.App = Nothing
End Sub
```

经过以上的设置，Excel 即可捕获 Application 对象的事件。需要注意的是，对于 Application 对象的事件过程代码必须编写在类模块 EventClassModule 中。

11.4.2 编写 Application 事件过程

使用前面介绍的方法编写类模块，可使 Application 对象响应用户的事件。下面以 WorkbookBeforeSave 事件为例演示使用方法。Application 对象的事件很多，有关其他事件的代码编写方法，可参照下面的方法进行。

Application 对象的 WorkbookBeforeSave 事件在保存项目之前发生。在该事件过程中编写代码，可禁止保存项目。

提示：若 Workbook 对象的 BeforeSave 事件过程中编写有代码，则将先触发 Workbook 对象的 BeforeSave 事件。

WorkbookBeforeSave 事件过程的结构如下：

```
Private Sub App_WorkbookBeforeSave(ByVal Wb As Workbook, ByVal SaveAsUI As
Boolean, Cancel As Boolean)

End Sub
```

该事件过程有 3 个参数，各参数的含义如下所述。

- pjb：表示要保存的工作簿。
- SaveAsUi：表示如果显示【另存为】对话框，则该值为 True。
- Cancel：表示事件发生时该参数的值为 False。如果事件过程将该参数设置为 True，则不会保存项目。

使用上面介绍的方法启用 Application 对象的事件后，在类模块 EventClassModule 的对象列表中选择 App，在事件列表中选择 WorkbookBeforeSave，将自动生成事件过程结构。在事件过程中编写以下代码，即可禁止保存项目。

```
Private Sub App_WorkbookBeforeSave(ByVal Wb As Workbook, ByVal SaveAsUI As
```

```
Boolean, Cancel As Boolean)
    MsgBox "本项作簿不允许保存修改内容！", vbCritical + vbOKOnly
    Cancel = True
End Sub
```

当保存工作簿时，将弹出如图 11-13 所示的提示框。

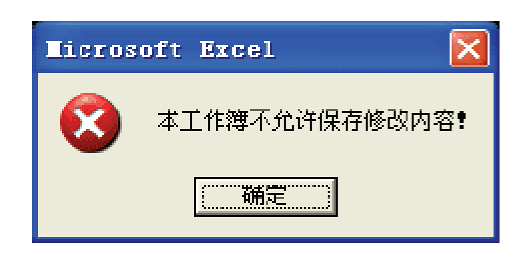


图 11-13　禁止保存工作簿

注意：如果在输入上面的代码之前已经启用 Application 对象的事件，则输入的代码也不能保存。为了能保存编写的该部分代码，须先运行“禁止 Application 事件”过程禁用 Application 对象的事件，将上面的事件代码保存后，再启用 Application 对象的事件。

第 12 章　使用 Workbook 对象

一个工作簿对象（Workbook）就是一个 Excel 文件，多个 Workbook 对象组成 Workbooks 集合。工作簿是 Excel 文档的基础，基于工作簿的代码主要有新建、打开、保存工作簿等相关代码。

12.1　了解 Workbook 对象

Workbook 工作簿对象位于 Application 对象的下一层次。对工作簿对象的操作就是对 Excel 文件的操作。本节先简单介绍 Workbook 对象常用的属性、方法和事件，在后面几节中将详细介绍使用这些属性、方法和事件控制工作簿的方法。

12.1.1　Workbooks 集合

在 Excel 中，每打开一个工作簿就增加了一个 Workbook 对象，这些 Workbook 对象都是 Workbooks 集合的一个元素。

可以使用 Workbooks 集合创建新的工作簿，关闭操作工作簿等。

Workbooks 集合具有集合对象所有的属性，例如 Count 属性返回集合中对象的数量等，这里不再介绍。Workbooks 集合对象提供以下方法，用来管理工作簿对象。

- ❑ Add 方法，新建工作簿。使用 Add 方法可以创建一个新的工作簿。Excel 会自动将该工作簿命名为 BookN，其中“N”是下一个可用的数字，新工作簿将成为活动工作簿。
- ❑ Close 方法，关闭工作簿。使用该方法，将关闭所有的 Excel 工作簿，但不退出 Excel 程序。如果某个打开的工作簿有改动，Excel 将显示询问是否保存更改的对话框和相应提示。
- ❑ Open 方法，打开工作簿。用 Open 方法打开一个工作簿时，该工作簿将成为 Workbooks 集合的成员。

另外，Workbooks 集合还提供了 OpenDatabase、OpenText 和 OpenXML 方法，分别用来打开数据库、文本文件和 XML 数据文件。

12.1.2　Workbook 常用属性

Workbook 对象提供了很多属性，使用这些属性可充分控制工作簿。下面介绍一些最常用的属性。

- ActiveSheet 属性：返回一个对象，它代表活动工作簿中或指定的窗口、工作簿中的活动工作表（最上面的工作表）。如果没有活动的工作表，则返回 Nothing。
- Application 属性：返回一个 Application 对象，该对象表示 Excel 应用程序。
- Charts 属性：返回一个 Sheets 集合，它代表指定工作簿中的所有图表工作簿。
- FullName 属性：返回对象的名称（用字符串表示），包括其磁盘路径。
- Name 属性：返回工作簿的文件名称，不含路径部分。
- Names 属性：返回一个 Names 集合，它代表指定工作簿的所有名称（Name）对象（包括所有指定工作表的名称）。每一个 Name 对象都代表一个单元格区域的定义名称。
- Password 属性：返回或设置在打开指定工作簿时必须提供的密码。
- Path 属性：返回应用程序的完整路径，不包括末尾的分隔符和应用程序名称。
- ReadOnly 属性：检查工作簿是否以只读方式打开。
- Saved 属性：如果指定工作簿从上次保存至今未发生过更改，则该属性值为 True。如果要关闭某个已更改的工作簿，但又不想保存它或者不想出现保存提示，则可将此属性设为 True。
- Sheets 属性：返回一个 Sheets 集合，它代表指定工作簿中的所有工作表。
- Worksheets 属性：返回一个 Sheets 集合，它代表指定工作簿中的所有工作表。
- Windows 属性：返回一个 Windows 集合，它代表指定工作簿中的所有窗口。

12.1.3　Workbook 常用方法

Workbook 提供了很多方法，下面介绍一些常用方法。

- Activate 方法：激活与工作簿相关的第一个窗口。
- Close 方法：关闭 Workbook 对象。
- Protect 方法：保护工作簿使其不被修改。这里只列出方法的作用。
- Save 方法：保存对指定工作簿所做的更改。
- SaveAs 方法：将工作簿换名保存。
- SaveCopyAs 方法：将指定工作簿的副本保存到文件，但不修改内存中的打开工作簿。使用该方法可以创建工作簿的备份，同时不修改原有工作簿的位置。
- Unprotect 方法：取消工作表或工作簿的保护。如果工作表或工作簿不是受保护的，则此方法不起作用。如果在保护工作簿时设置了密码，则取消保护时也需要提供密码。

12.1.4　Workbook 常用事件

Workbook 提供了丰富的事件接口，方便用户编写程序对 Workbook 对象进行控制。下面介绍一些常用的事件。

- BeforeClose 事件：在关闭工作簿之前，先产生此事件。如果该工作簿已经更改过，则此事件在询问用户是否保存更改之前产生。
- BeforePrint 事件：在打印指定工作簿（或者其中的任何内容）之前，发生此事件。

- ❑ NewSheet 事件：当在工作簿中新建工作表时发生此事件。
- ❑ Open 事件：打开工作簿时，发生此事件。
- ❑ SheetActivate 事件：当激活任何工作表时发生此事件。
- ❑ SheetDeactivate 事件：该事件与 SheetActivate 事件作用相反，当任何工作表被停用时发生此事件。
- ❑ WindowActivate 事件：工作簿窗口被激活时，将发生此事件。
- ❑ WindowDeactivate 事件：任何工作簿窗口被停用时将发生此事件。
- ❑ WindowResize 事件：任何工作簿窗口调整大小时将发生此事件。

12.2　控制工作簿集合

多个 Workbook 对象组成 Workbooks 集合。对工作簿集合的操作包括新建、打开、保存工作簿等内容。

12.2.1　新建工作簿

Workbooks 集合中包含了 Excel 应用程序当前打开的所有 WorkBook 对象。可以使用 Workbooks 集合创建新的工作簿，关闭操作工作簿等。

在 VBA 中创建新的工作簿，可以使用 Workbooks 集合对象的 Add 方法。下面的代码创建一个新的工作簿。Excel 自动将该工作簿命名为 BookN，其中“N”是下一个可用的数字。新工作簿将成为活动工作簿。

```
Sub AddWorkbook()
    Workbooks.Add
End Sub
```

创建新工作簿更好的方法是将其分配给一个对象变量，在程序中可以通过该对象变量对工作簿进行设置。使用对象变量可以很容易地控制新工作簿。

例如，以下代码就可完成创建新的工作簿，并设置工作簿的相关属性。

```
Sub AddNew()
    n=Workbooks.Count
    Set NewBook = Workbooks.Add

    With NewBook
        .Title = "新工作簿" & n
        .SaveAs Filename:="新工作簿" & n & ".xls"
    End With
End Sub
```

12.2.2　打开工作簿

使用 Workbooks 集合对象的 Open 方法打开一个工作簿时，该工作簿将成为 Workbooks

集合的成员。下述代码将打开“D:\Excel 工作簿\使用 Workbook 对象.xls”工作簿。

```
Sub OpenWorkbook()
    Workbooks.Open ("D:\Excel 工作簿\使用 Workbook 对象.xls")
End Sub
```

在更多的时候，打开工作簿时需要查找文件所在位置，这时可通过【打开】对话框来进行查找。显示【打开】对话框的语法格式如下：

```
Application.GetOpenFilename(FileFilter, FilterIndex, Title, ButtonText,
MultiSelect)
```

使用 GetOpenFilename 方法返回选定的文件名或用户输入的名称。返回的名称可能包含路径说明。如果用户取消了对话框，则该值为 False。例如，以下代码要求用户选择一个工作簿，并使用 Open 方法将其打开。

```
Sub 打开工作簿()
    Dim fm As String, flag As Boolean

    flag = False
    Do While Not flag    '对话框打开已有 Excel 文件
        fm = Application.GetOpenFilename(fileFilter:="Excel files(*.xls),*.
        xls, All files (*.*),*.*")
        If fm <> "False" Then
            Workbooks.Open fm
            Set bb = ActiveWorkbook
            flag = True
        End If
    Loop
End Sub
```

以上代码首先通过 GetOpenFilename 方法显示【打开】对话框，如图 12-1 所示。如果用户没有选择要打开的工作簿（单击【取消】按钮），则将反复显示【打开】对话框。

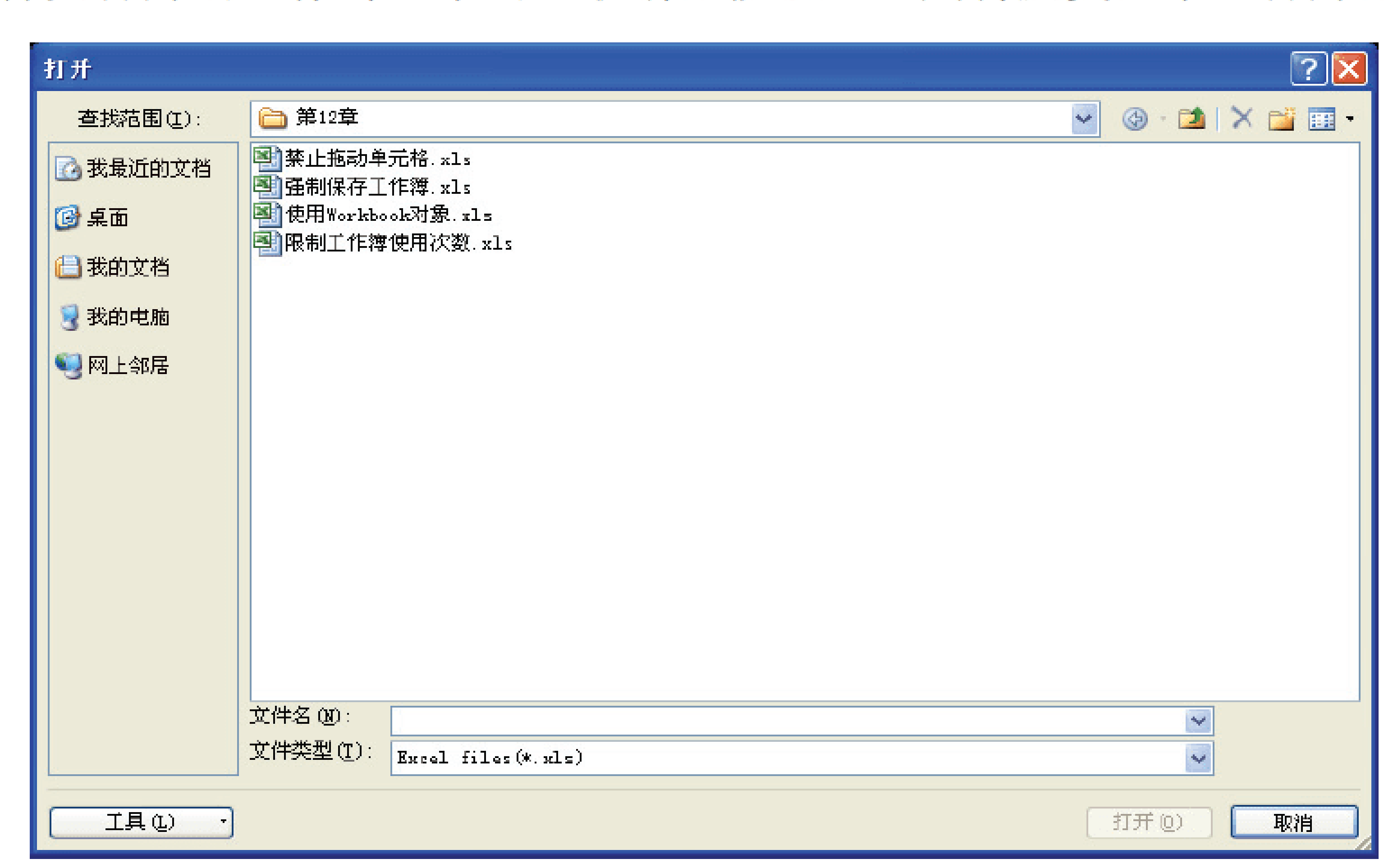


图 12-1　打开工作簿

12.2.3 打开文本文件

使用 Workbooks 集合对象的 OpenText 方法，可打开一个文本文件，并将其作为包含单个工作表的新工作簿进行分列处理，然后在此工作表中放入经过分列处理的文本文件数据。该方法的语法格式如下：

```
Application.OpenText(Filename, Origin, StartRow, DataType, TextQualifier,
ConsecutiveDelimiter, Tab, Semicolon, Comma, Space, Other, OtherChar,
FieldInfo, TextVisualLayout, DecimalSeparator, ThousandsSeparator,
TrailingMinusNumbers, Local)
```

该方法的参数很多，除了 Filename 为必需的参数之外，其他参数都可省略。各参数的含义如下所述。

- ❑ Filename：指定要打开和分列的文本文件的名称。
- ❑ Origin：指定文本文件来源。可为以下常量 xlMacintosh，xlWindows 或 xlMSDOS。此外，它还可以是一个整数，表示所需代码页的代码页编号。例如，“1256”指定源文本文件的编码是阿拉伯语。如果省略该参数，则此方法将使用“文本导入向导”中“文件原始格式”选项的当前设置。
- ❑ StartRow：文本分列处理的起始行号。默认值为 1。
- ❑ DataType：指定文件中数据的列格式。可为常量 xlDelimited 或 xlFixedWidth。如果未指定该参数，则 Excel 将尝试在打开文件时确定列格式。
- ❑ TextQualifier：指定文本识别符号。
- ❑ ConsecutiveDelimiter：如果为 True，则将连续分隔符视为一个分隔符。默认值为 False。
- ❑ Tab：如果为 True，则将制表符用作分隔符（DataType 必须为 xlDelimited）。默认值为 False。
- ❑ Semicolon：如果为 True，则将分号用作分隔符（DataType 必须为 xlDelimited）。默认值为 False。
- ❑ Comma：如果为 True，则将逗号用作分隔符（DataType 必须为 xlDelimited）。默认值为 False。
- ❑ Space：如果为 True，则将空格用作分隔符（DataType 必须为 xlDelimited）。默认值为 False。
- ❑ Other：如果为 True，则将 OtherChar 参数指定的字符用作分隔符（DataType 必须为 xlDelimited）。默认值为 False。
- ❑ OtherChar：（如果 Other 为 True，则为必选项）。当 Other 为 True 时，指定分隔符。如果指定了多个字符，则仅使用字符串中的第一个字符而忽略剩余字符。
- ❑ FieldInfo：包含单列数据相关分列信息的数组。对该参数的解释取决于 DataType 的值。如果此数据由分隔符分隔，则该参数为由两元素数组组成的数组，其中每个两元素数组指定一个特定列的转换选项。第一个元素为列标（从 1 开始），第二个元素是 XlColumnDataType 的常量之一，用于指定分列方式。

- ❑ TextVisualLayout：文本的可视布局。
- ❑ DecimalSeparator：识别数字时，Excel 使用的小数分隔符。默认设置为系统设置。
- ❑ ThousandsSeparator：识别数字时，Excel 使用的千位分隔符。默认设置为系统设置。
- ❑ TrailingMinusNumbers：如果应将结尾为减号字符的数字视为负数处理，则指定为 True。如果为 False 或省略该参数，则将结尾为减号字符的数字视为文本处理。
- ❑ Local：如果分隔符、数字和数据格式应使用计算机的区域设置，则指定为 True。

例如，以下代码将打开文本文件“员工花名册.txt”：

```
Sub 打开文本文件()
    Workbooks.OpenText Filename:="员工花名册.txt", _
        DataType:=xlDelimited, Tab:=True
End Sub
```

执行以上代码，将文本文件“员工花名册.txt”打开，如图 12-2 所示。

	A	B	C	D	E	F	G	H
1	编号	姓名	身份证号	性别	民族	出生年月	学历	通信地址
2	Y0001	伍云辉	4.3E+17	男	汉	########	本科	红旗路69号
3	Y0002	王大华	5.1E+17	男	汉	1980-1-1	本科	胜利路28号
4	Y0003	张小山	3.3E+18	男	汉	1978-8-8	专科	西环路104号
5	Y0004	李萍	1.01E+18	女	汉	1981-5-5	本科	四川路33号
6	Y0005	王涛	3.3E+18	男	汉	1970-8-8	专科	新达路3号
7	Y0006	罗建	51010119	男	汉	1983-1-1	专科	万达路99号

图 12-2　打开的文本文件

从图中可看出，打开的文本文件已导入到 Excel 工作簿中，该工作簿只有一个工作表，工作表名称与文本文件名称相同。

提示： 在图 12-2 的工作表中，“身份证号”是按数值类型保存，显示为科学计数方式。因此，直接打开文本文件时，对于这类长数值数据，其精度将受影响。

12.2.4　工作簿是否存在

在 VBA 程序中，如果打开一个不存在的工作簿，将产生错误并中断程序的执行。因此，在打开工作簿之前，先判断该文件是否存在是很有必要的。

可以使用 VBA 的 Dir 函数来检查某些文件或目录是否存在。其语法格式如下：

```
Dir[(pathname[, attributes])]
```

该函数的两个参数都可省略，其中，pathname 用来指定文件名的字符串表达式，可能包含文件夹、驱动器。如果没有找到 pathname，则会返回零长度字符串（""）。Attributes 是一个常数或数值表达式，其总和用来指定文件属性。如果省略，则会返回匹配 pathname 但不包含属性的文件。

例如，使用以下代码将判断工作簿是否存在：

```
Sub 工作簿是否存在()
    Dim str1 As String

    str1 = Application.InputBox(prompt:="请输入 Excel 工作簿文件名: ", _
        Title:="文件名", Type:=2)

    If str1 = "False" Then Exit Sub
    str1 = ActiveWorkbook.Path & "\" & str1
    If Not FileExists(str1) Then
        MsgBox "工作簿“" & str1 & "”不存在!"
    Else
        Workbooks.Open str1
    End If
End Sub
```

在上面的代码中，用到一个自定义函数 FileExists，该函数用来判断指定文件是否存在。其代码如下：

```
Function FileExists(FullFileName As String) As Boolean
    '如果工作簿存在,则返回 True
    FileExists = Len(Dir(FullFileName)) > 0
End Function
```

上面的代码使用 Dir 函数判断指定文件是否存在。使用 Dir 函数还可以方便地获取指定路径下的文件夹或所有文件名。

执行以上过程“工作簿是否存在”，将弹出如图 12-3 所示的对话框，提示用户输入工作簿名称（不用输入路径和扩展名），单击【确定】按钮关闭对话框。若输入的工作簿存在，则调用 Open 方法打开工作簿；否则，显示如图 12-4 所示的提示对话框。



图 12-3　输入文件名



图 12-4　提示对话框

12.2.5　工作簿是否打开

在 Workbook 对象中没有现成的属性或方法判断工作簿是否打开。这时，可使用 VBA 的错误捕捉功能来判断工作簿是否打开。如果工作簿已经打开，可使用 VBA 的以下代码引用具体的工作簿名称：

```
Set wb = Workbooks(工作簿名称)
```

如果引用的工作簿名称不存在，VBA 将产生一个错误信息。

使用 On Error 语句捕获错误，如果产生错误信息，则 Err 对象的 Number 属性值将返回错误值。否则，Err 对象的 Number 属性值为 0。

例如，使用以下代码可判断工作簿是否打开：

```
Sub 工作簿是否打开()
    Dim str1 As String
    str1 = Application.InputBox(prompt:="请输入 Excel 工作簿文件名: ", _
        Title:="文件名", Type:=2)
    If str1 = "False" Then Exit Sub
    If Not IsOpen(str1) Then
        MsgBox "工作簿“" & str1 & "”未打开!"
    Else
        MsgBox "工作簿“" & str1 & "”已打开!"
    End If
End Sub
```

以上程序调用了一个自定义函数 IsOpen，该函数的 VBA 代码如下：

```
Private Function IsOpen(WorkBookName As String) As Boolean
    '如果该工作簿已打开则返回真
    Dim wb As Workbook
    On Error Resume Next
    Set wb = Workbooks(WorkBookName)
    If Err = 0 Then
        IsOpen = True
    Else
        IsOpen = False
    End If
End Function
```

执行以上过程“工作簿是否打开”，将弹出如图 12-5 所示的对话框，提示用户输入工作簿名称（需要输入扩展名），单击【确定】按钮关闭对话框。若输入的工作簿已打开，则将显示如图 12-6 所示的提示对话框。



图 12-5　输入文件名



图 12-6　提示对话框

12.3　控制工作簿

Workbook 对象也提供了许多的属性和方法，以控制工作簿。通过工作簿的属性可获取对当前工作簿各工作表的引用，这些内容将在下一章中进行介绍。本节介绍对工作簿的保存、获取文档属性等操作。

12.3.1　保存工作簿

保存工作簿时，需使用 Workbook 对象的 Save 方法，该方法将工作簿的修改保存到磁

盘中，不需要任何参数。

提示：要将工作簿标记为已保存，但不将其写入磁盘，可将工作簿的 Saved 属性设置为 True。首次保存工作簿时，应使用 SaveAs 方法指定文件名。

例如，以下代码对当前所有打开的工作簿进行判断，如果为新建工作簿，则调用 Save 方法保存：

```
Sub 保存新建工作簿()
    Dim wb1 As Workbook
    For Each wb1 In Workbooks
        If wb1.Path <> "" Then wb1.Save
    Next
End Sub
```

以上代码根据工作簿的 Path 参数来判断是否为新建工作簿，若 Path 属性值不为空，则保存该工作簿。

12.3.2 更名保存工作簿

保存工作簿有两种方式：一种是保存修改到原工作簿中；另一种是保存工作簿的另一个副本。

使用 SaveAs 方法可将工作簿更名保存。例如，以下代码新建一个工作簿，提示用户输入文件名，保存该工作簿。

```
Set NewBook = Workbooks.Add
Do
    fName = Application.GetSaveAsFilename
Loop Until fName <> False
NewBook.SaveAs Filename:=fName
```

12.3.3 设置工作簿密码

为了安全性，有时需要为工作簿设置打开权限密码。使用 Workbook 对象的 Password 属性可获取或设置该密码。

例如，以下代码将提示用户为工作簿设置一个密码，然后保存退出。

```
Sub 设置密码()
    ActiveWorkbook.Password = InputBox("输入密码：")
    ActiveWorkbook.Close
End Sub
```

执行以上代码后，将关闭当前 Excel 工作簿。下次打开此工作簿时，将弹出如图 12-7 所示的【密码】对话框，要求用户输入正确的密码才能打开工作簿。

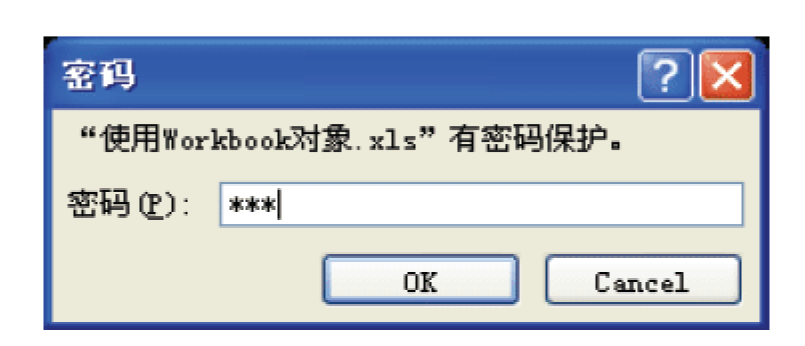


图 12-7　输入密码

要取消工作簿的密码，可设置其 Password 属性为空字符串，代码如下：

```
Sub 取消密码()
    ActiveWorkbook.Password = ""
End Sub
```

取消工作簿的密码也可通过单击【Office 按钮】打开下拉菜单，从下拉菜单中选择【准备】|【加密文档】命令，打开如图 12-8 所示的【加密文档】对话框，删除【密码】文本框中的字符即可。

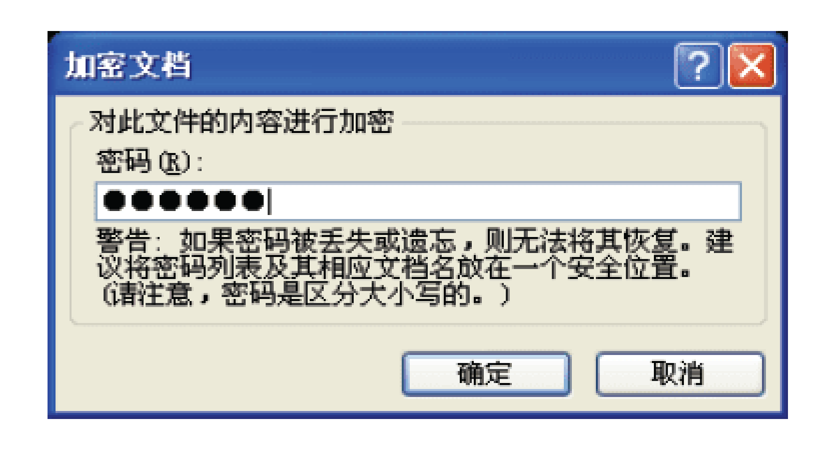


图 12-8 【加密文档】对话框

12.3.4　查看文档属性

在 Excel 中，要查看和修改文档的属性可按以下步骤操作：

（1）单击【Office 按钮】，打开下拉菜单。

（2）单击主菜单【准备】|【属性】命令，在 Excel 的功能区下方将打开如图 12-9 所示的【文档属性】面板，供用户查看和修改。

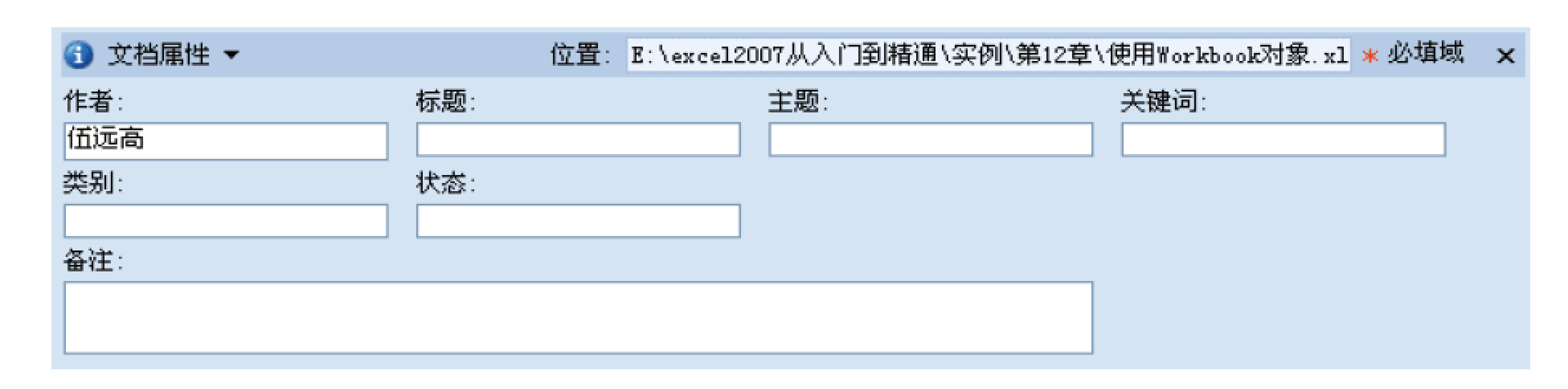


图 12-9 【文档属性】面板

（3）单击左上角【文档属性】标签，打开下拉菜单，选择【高级属性】命令，打开如图 12-10 所示的属性窗口，在其【自定义】选项卡中可定义各种属性值。



图 12-10　自定义属性

以上是在 Excel 操作环境下操作的结果。

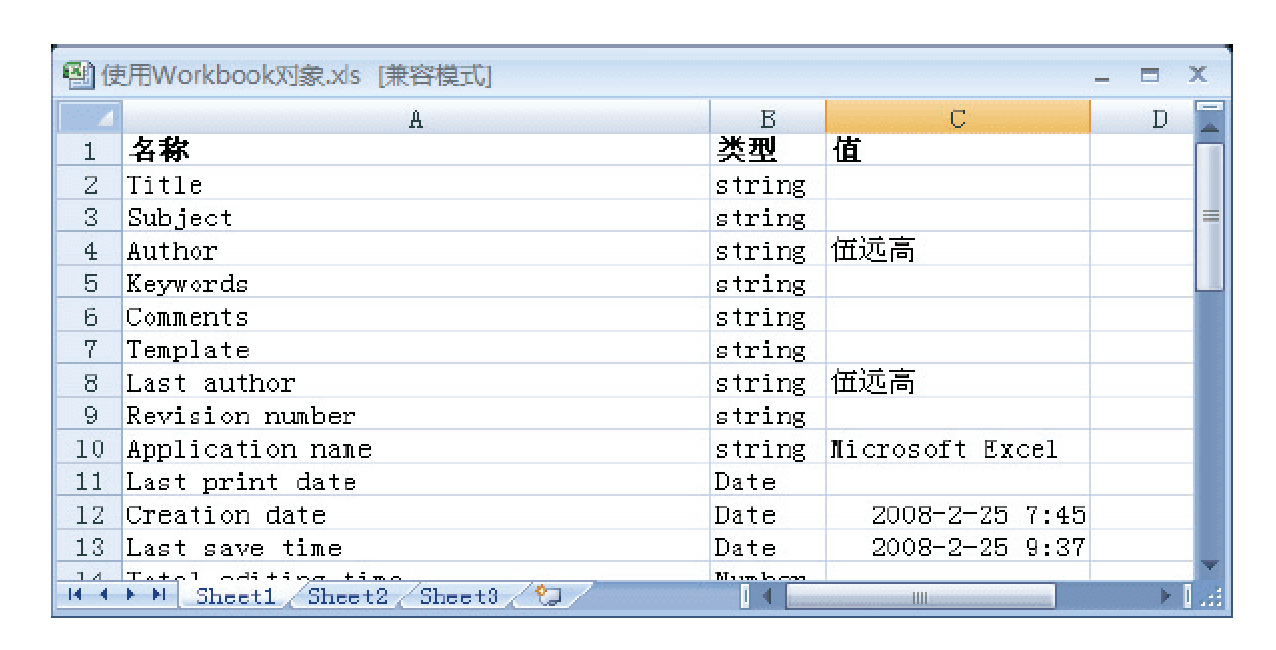
在 Excel 程序开发中，也可使用 VBA 代码控制并设置文档的属性值。使用 Workbook 对象的 BuiltinDocumentProperties 属性返回一个 DocumentProperties 集合，该集合表示指定工作簿的所有内置文档属性。

例如，以下代码通过在 BuiltinDocumentProperties 属性的 DocumentProperties 集合中逐个读出属性，并将其属性名称和属性值填写到 Sheet1 工作表的第 A、B 两列中。

```
Sub 文档属性()
    Dim r As Integer
    Worksheets(1).Activate
    Cells(1, 1) = "名称"
    Cells(1, 2) = "类型"
    Cells(1, 3) = "值"
    Range("A1:C1").Font.Bold = True
    With ActiveWorkbook
        For r = 1 To .BuiltinDocumentProperties.Count
            With .BuiltinDocumentProperties(r)
                Cells(r + 1, 1) = .Name
                Select Case .Type
                Case msoPropertyTypeBoolean
                    Cells(r + 1, 2) = "Boolean"
                Case msoPropertyTypeDate
                    Cells(r + 1, 2) = "Date"
                Case msoPropertyTypeFloat
                    Cells(r + 1, 2) = "Float"
                Case msoPropertyTypeNumber
                    Cells(r + 1, 2) = "Number"
                Case msoPropertyTypeString
                    Cells(r + 1, 2) = "string"
                End Select
                On Error Resume Next
                Cells(r + 1, 3) = .Value
```

```
            On Error GoTo 0
        End With
    Next r
  End With
  Range("A:C").Columns.AutoFit
End Sub
```

程序运行结果如图 12-11 所示。



	A	B	C
1	名称	类型	值
2	Title	string	
3	Subject	string	
4	Author	string	伍远高
5	Keywords	string	
6	Comments	string	
7	Template	string	
8	Last author	string	伍远高
9	Revision number	string	
10	Application name	string	Microsoft Excel
11	Last print date	Date	
12	Creation date	Date	2008-2-25 7:45
13	Last save time	Date	2008-2-25 9:37

图 12-11 文档属性

12.3.5 处理工作簿文件名

使用 Workbook 对象的 FullName 属性可返回工作簿的名称（用字符串表示），包括其磁盘路径。而 Name 属性将只返回工作簿文件名，不包括磁盘路径。

例如，以下代码将显示当前活动工作簿的文件名和全路径名。

```
Sub 获取文件名()
  MsgBox "当前工作簿名称为：" & ActiveWorkbook.Name & vbNewLine & _
    "当前工作簿全路径名为：" & ActiveWorkbook.FullName
End Sub
```

执行以上代码可显示如图 12-12 所示的对话框。

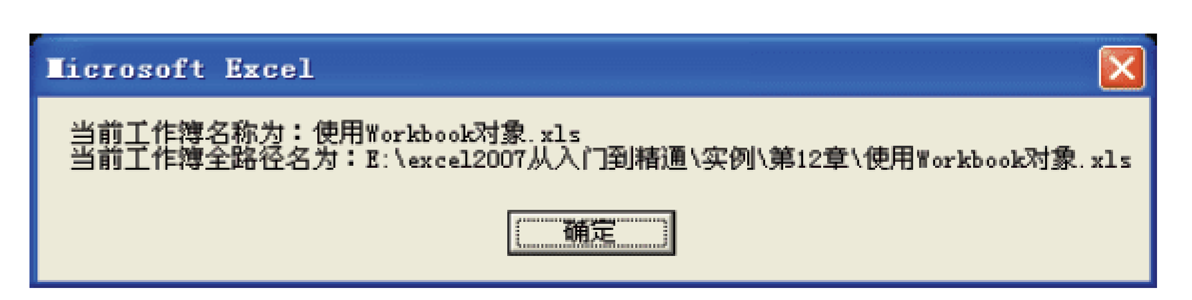


图 12-12 显示工作簿文件名

12.4 响应用户的动作

在 VBE 左侧的【工程】资源管理器窗口中，Excel 对象列表中除了显示每个工作簿之外，还有一个名为 ThisWorkbook 的对象，该对象表示其中正在运行当前宏代码的工作簿。

双击该对象打开代码窗口，在此代码窗口中编写工作簿事件过程代码。通过工作簿事件可控制工作簿的打开、关闭、打印等操作，此外，工作簿事件中还提供了控制工作表的很多事件。

12.4.1 自动打开关联工作簿

在 Excel 工作簿中，有时需要引用另一个工作簿中的数据。对于这样两个有关联的工作簿，在打开主工作簿时可同时打开关联工作簿。

打开工作簿时，将产生 Open 事件，可在该事件中编写代码打开关联工作簿。该事件过程结构如下：

```
Private Sub Workbook_Open()

End Sub
```

例如，以下代码在打开主工作簿时，将同时打开名为“关联工作簿.xls”的文件。

```
Private Sub Workbook_Open()
    Workbooks.Open (ThisWorkbook.Path & "\关联工作簿.xls")
End Sub
```

12.4.2 禁止拖动单元格

在正常情况下，将鼠标移动到 Excel 工作表单元格边缘时，鼠标指针将变为十字箭头状，这时按下鼠标左键拖动，可移动单元格的数据。在有的情况下，需要禁止这种拖动方式移动数据。

通过 Application 对象的 CellDragAndDrop 属性可控制单元格拖放功能。如果启用单元格拖放功能，则该属性值为 True。

若需要在整个工作簿打开期间都禁止拖放功能，可在工作簿的 Open 事件中编写代码，设置 CellDragAndDrop 属性为 False。具体代码如下：

```
Private Sub Workbook_Open()
    Application.CellDragAndDrop = False
End Sub
```

在退出工作簿时，应该将 CellDragAndDrop 属性设置为 True，恢复拖放功能。这时可在 BeforeClose 事件中编写代码。在关闭工作簿之前，先产生 BeforeClose 事件。如果该工作簿已经更改过，则本事件在询问用户是否保存更改之前产生。其事件过程的结构如下：

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)

End Sub
```

参数 Cancel 在事件发生时为 False。如果该事件过程将此参数设置为 True，则停止关闭操作，工作簿保持打开状态。

在工作簿对象 Workbook 的 BeforeClose 事件过程中编写 VBA 代码，在关闭工作簿之前允许单元格拖放功能。

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    Application.CellDragAndDrop = True
End Sub
```

12.4.3　退出前强制保存工作簿

在 Excel 工作簿中进行了编辑操作，如果未进行保存操作就关闭工作簿，Excel 将弹出如图 12-13 所示的提示窗口，让用户选择是否保存对工作簿的更改，该对话框共有 3 个选项：

- 单击【是】按钮，保存更改并退出工作簿；
- 单击【否】按钮，不保存并退出工作簿；
- 单击【取消】按钮，不保存工作簿，并返回工作簿继续操作。



图 12-13　保存工作簿提示对话框

在用 Excel 开发一些应用系统时，如果希望在关闭工作簿时不弹出如图 12-13 所示的对话框而自动保存工作簿。这时，可在 BeforeClose 事件过程中编写代码。在关闭 Excel 工作簿之前，将先产生 BeforeClose 事件。该事件过程的结构如下所示：

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)

End Sub
```

参数 Cancel 为一个逻辑变量，当进入事件过程时，该参数的值为 False。如果在事件过程中将此参数设置为 True，则停止关闭操作，返回到工作簿中继续操作（工作簿保持打开状态）。

例如，在 ThisWorkbook 对象的 BeforeClose 事件代码中编写如下代码：

```
Private Sub Workbook_BeforeClose(Cancel as Boolean)
    If Me.Saved = False Then Me.Save
End Sub
```

以上代码根据 Workbook 对象的 Saved 属性来判断工作簿是否保存。若 Saved 属性值为 False（表示工作簿的数据经过修改还未保存），则调用 Workbook 对象的 Save 方法保存工作簿。

如果指定工作簿从上次保存一直未发生过更改，则该 Saved 属性值为 True。

如果要关闭某个已更改的工作簿，但又不想保存它或者不想出现保存提示，则可将

Saved 属性设为 True。

提示：如果某个工作簿从未保存过，则其 Path 属性返回一个空字符串（""）。

12.4.4 禁止保存工作簿

在第 11 章介绍 Application 对象事件时，介绍了禁止用户保存工作簿的代码。该代码在 Application 对象的 WorkbookBeforeSave 事件过程中编写。

其实，要实现该功能更好的方法是在 Workbook 对象的 BeforeSave 事件过程中编写代码。Workbook 对象的 BeforeSave 事件将比 Application 对象的 WorkbookBeforeSave 事件过程先触发。

在保存工作簿之前将产生 BeforeSave 事件，其事件过程结构如下：

```
Private Sub Workbook_BeforeSave(ByVal SaveAsUi As Boolean, Cancel As Boolean)

End Sub
```

该事件过程有两个参数，其含义如下所述。

- SaveAsUI：若设为 True，将显示【另存为】对话框。
- Cancel：当事件发生时该参数的值为 False。如果该事件过程将此参数设置为 True，则该过程完成后将不保存工作簿。

针对该事件的两个参数，该事件一般可以完成以下功能。

- 禁止文件另存，但可对原文件的修改进行保存；
- 禁止保存修改，使保存与另存为功能都失效。

在禁止保存修改时，应配合在 BeforeClose 事件中编写以下代码才能达到完美效果；否则退出 Excel 时将弹出保存提示对话框。

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    Me.Saved=True
End Sub
```

在工作簿的 BeforeSave 事件中编写以下 VBA 代码，禁止 Excel 的“保存”功能。

```
Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)
    Cancel = True                                   '禁止保存修改
    MsgBox "本工作簿不允许保存修改内容！", vbCritical + vbOKOnly
End Sub
```

若要禁止“另存为”功能，可使用以下代码：

```
Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)
    If SaveAsUI = True Then Cancel = True    '禁止另存为
    MsgBox "本工作簿不允许保存修改内容！", vbCritical + vbOKOnly
End Sub
```

12.4.5　限制工作簿使用次数

有的工作簿需要限制用户使用的次数。例如，应用程序的演示版允许用户使用 10 次，在正常情况下不会出现提示信息。当使用次数超过 7 次时，将显示还能使用的次数，如图 12-14 所示。当使用次数达到 10 次后，再次打开本例工作簿时将自动删除本工作簿，如图 12-15 所示。

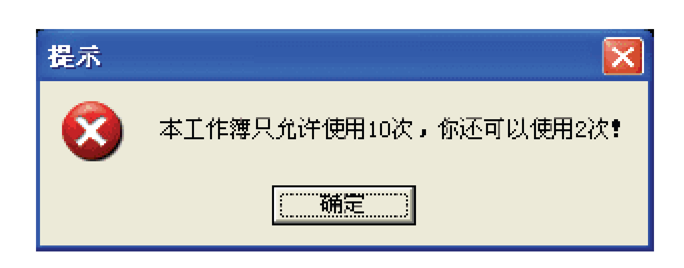



图 12-14　显示使用次数提示

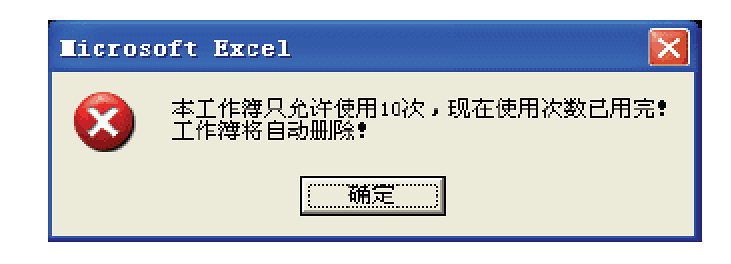


图 12-15　超过使用次数提示

要实现这样的功能，可在 Workbook 对象的 Open 事件中编写代码，在打开工作簿时检查使用的次数。具体代码如下：

```
Private Sub Workbook_Open()
    Dim t As Integer
    t = ActiveSheet.Cells(1, 255).Value
    t = t + 1
    ActiveSheet.Cells(1, 255) = t                        '保存使用次数
    If t > 7 And t <= 10 Then
        MsgBox "本工作簿只允许使用 10 次，现在还可以使用" & 10 - t & "次！", _
            vbCritical + vbOKOnly, "提示"
    ElseIf t > 10 Then
        MsgBox "本工作簿只允许使用 10 次，现在使用次数已用完！" & _
            vbNewLine & "工作簿将自动删除！", _
            vbCritical + vbOKOnly
        ActiveWorkbook.ChangeFileAccess xlReadOnly   '更改工作簿的访问权限
        Kill ActiveWorkbook.FullName                 '删除工作簿
        Me.Saved = True                              '修改更改状态
        Application.Quit                             '退出 Excel
    End If
End Sub
```

在工作簿中，活动工作表的第 1 行第 255 列保存的是已使用次数。当打开工作簿时从该单元格读出数据进行判断，若超过 10 次，则使用 Kill 语句删除当前工作簿。因当前工作簿处于打开状态，直接执行 Kill 语句将会产生错误，所以还需使用 ChangeFileAccess 方法更改工作簿的访问权限，该方法的语法格式如下：

```
表达式.ChangeFileAccess(Mode, WritePassword, Notify)
```

以上参数中 Mode 为必需的，其余两个参数可省略。各参数的含义如下所述。

- Mode：为工作簿指定新的访问模式，可设置为 xlReadOnly（只读）或 xlRead/Write（可读/写）两种模式。
- WritePassword：如果文件设置了写保护并且 Mode 为 xlReadWrite，则指定写保护

密码。如果文件没有密码或 Mode 为 xlReadOnly，则忽略此参数。

- Notify：如果该值为 True（或省略该参数），则当无法立即访问文件时通知用户。

如果以只读模式打开文件，则不可独占访问此文件。如果将此文件从只读更改为可读写，Excel 必须载入该文件的新副本以确认在以只读模式打开该文件后没有进行过更改。

为了使工作簿退出时保存使用次数，在工作簿对象 Workbook 对象的 BeforeClose 事件过程中编写以下代码，在关闭 Excel 工作簿之前自动保存。

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    If Me.Saved = False Then Me.Save
End Sub
```

12.4.6 限制打印

在如图 12-16 所示的【Office 按钮】下拉菜单中，选择【打印】选项卡，可打印工作簿中当前工作表或工作簿中的所有工作表。

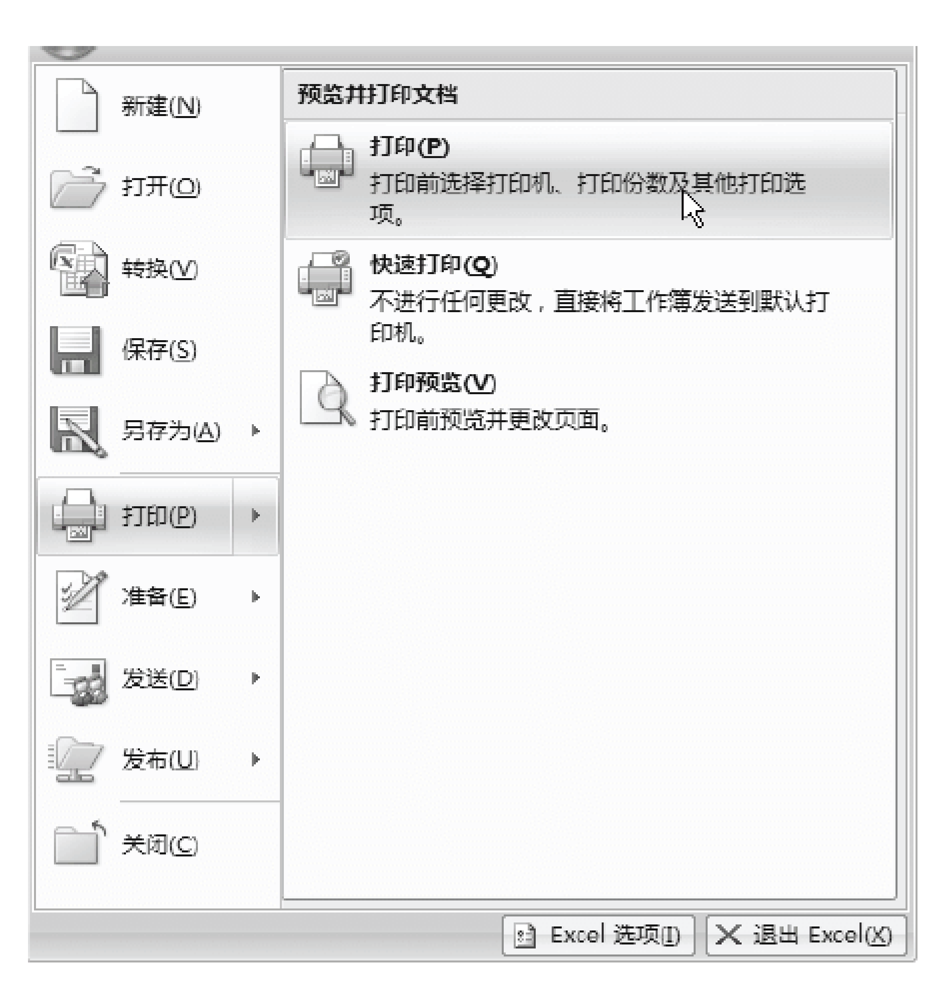


图 12-16 【打印】选项卡

在某些情况下，工作表中的数据将只允许用户查看，不允许打印。要实现这样的功能，可以通过 Workbook 对象的 BeforePrint 事件进行控制。在打印指定工作簿（或者其中的任何内容）之前，将产生 BeforePrint 事件。该事件过程的结构如下所示：

```
Private Sub Workbook_BeforePrint(Cancel As Boolean)

End Sub
```

参数 Cancel 为逻辑型，当事件发生时其值为 False。如果该事件过程将此参数设置为 True，则该过程完成后将不打印工作簿。

限制打印工作簿的代码很简单，只需要在 Workbook 事件中的 BeforePrint 事件过程中编写如下代码即可。

```
Private Sub Workbook_BeforePrint(Cancel As Boolean)
    MsgBox "本工作簿的内容不允许打印！", vbCritical + vbOKOnly
    Cancel = True
End Sub
```

在 Workbook 对象的 BeforePrint 事件过程中编写以上代码后，当用户执行【打印】命令时，将弹出如图 12-17 所示的提示对话框，提示用户不允许打印数据。

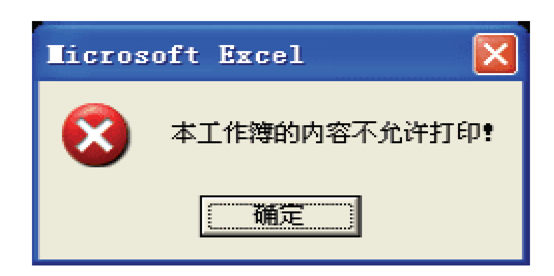


图 12-17　提示对话框

第 13 章　使用 Worksheet 对象

Worksheet 对象表示 Excel 工作表，通过 Workbook 对象的 Sheets 属性或 Worksheets 属性可返回指定工作簿中的工作表。

本章将介绍 Worksheets 集合对象和 Worksheet 对象的常用属性、方法和事件。

13.1　了解 Worksheet 对象

在 VBA 中，通过 Worksheets 集合对象可向工作簿中增加、删除工作表，获取对工作表的引用，向工作表中增加、删除行。通过 Worksheet 对象的事件还可控制工作表的行为，如禁止更改名称、禁止打印输出等。

13.1.1　Worksheets 集合

Worksheets 集合包括指定工作簿中的所有 Worksheet 对象。每个 Worksheet 对象都代表一个工作表。

提示：在 Excel 中还提供了一个 Sheets 集合，该集合中的每个成员都是 Worksheet 对象或 Chart 对象，其属性和方法都相同。

在 Worksheets 集合中，除了一般集合对象具有的属性外，还有一个 Visible 属性，通过该属性可控制集合中的 Worksheet 对象是否可见。

通过 Worksheets 集合的方法，可以对工作表进行控制，下面介绍几种常用的方法。

- Add 方法：新建工作表、图表或宏表。新建的工作表将成为活动工作表。
- Copy 方法：将工作表复制到工作簿的另一位置。
- Delete 方法：删除集合中的指定对象——工作表（Worksheet）对象。
- Move 方法：将工作表移到工作簿中的其他位置。
- PrintOut 方法：打印输出工作表。
- PrintPreview 方法：按对象打印后的外观效果显示对象的预览。

13.1.2　Worksheet 对象的常用属性

通过 Worksheet 对象的属性，可引用工作表中的单元格、处理批注、控制工作表可见性等。Worksheet 对象的常用属性如下所述。

- ❑ Cells 属性：返回一个 Range 对象，它代表工作表中的所有单元格（不仅仅是当前使用的单元格）。
- ❑ Comments 属性：返回一个 Comments 集合，该集合表示指定工作表的所有注释。
- ❑ Name 属性：返回或设置一个 String 值，它代表工作表的名称。
- ❑ Next 属性：返回代表下一个工作表的 Worksheet 对象。
- ❑ Previous 属性：返回代表上一个工作表的 Worksheet 对象
- ❑ Range 属性：返回一个 Range 对象，它代表一个单元格或单元格区域。
- ❑ ScrollArea 属性：以 A1 样式的区域引用形式返回或设置允许滚动的区域。用户不能选定滚动区域之外的单元格。
- ❑ UsedRange 属性：返回一个 Range 对象，该对象表示指定工作表上所使用的区域。
- ❑ Visible 属性：设置工作表对象是否可见。

13.1.3　Worksheet 对象的常用方法

使用 Worksheet 对象提供的方法，可复制、删除、移动工作表，还可对工作表进行保护等操作，Worksheet 对象的常用方法如下所示。

- ❑ Activate 方法：使当前工作表成为活动工作表。调用此方法等同于单击工作表的标签。
- ❑ Copy 方法：将工作表复制到工作簿的另一位置。
- ❑ Delete 方法：删除工作表对象。在删除工作表时，此方法显示一个对话框，用于提示用户确认是否删除，如果用户在对话框中单击【取消】按钮，则返回 False，如果用户单击【删除】按钮，则返回 True。
- ❑ Move 方法：将工作表移到工作簿中的其他位置。
- ❑ Paste 方法：将剪贴板中的内容粘贴到工作表上。
- ❑ Protect 方法：保护工作表使其不能被修改。
- ❑ Unprotect 方法：取消工作表的保护。如果工作表不是受保护的，则此方法不起作用。

13.1.4　Worksheet 对象的常用事件

通过 Worksheet 对象的事件，可捕获用户在工作表中的操作，以控制工作表中的数据。Worksheet 对象的常用事件如下所示。

- ❑ Activate 事件：激活工作表、图表工作表或嵌入式图表时发生此事件。
- ❑ BeforeDoubleClick 事件：当双击工作表时发生此事件，此事件先于默认的双击操作。
- ❑ BeforeRightClick 事件：右击工作表时发生此事件，此事件先于默认的右击操作。
- ❑ Calculate 事件：在对工作表进行重新计算之后发生此事件。
- ❑ Change 事件：当用户更改工作表中的单元格，或外部链接引起单元格的更改时发生此事件。

- ❑ Deactivate 事件：图表、工作表被停用时发生此事件。
- ❑ SelectionChange 事件：当工作表上的选定区域发生改变时发生此事件。

13.2　管理工作表

Worksheet 对象是 Worksheets 集合对象的成员，同时也是 Sheets 集合的成员。Sheets 集合包含工作簿中所有的工作表（图表工作表和工作表）。图表工作表的使用将在第 15 章中进行介绍，本章主要介绍工作表的相关操作。

13.2.1　新增工作表

使用 Worksheets 集合对象的 Add 方法，可向指定工作簿中增加工作表，该方法的语法格式如下：

```
表达式.Add(Before, After, Count, Type)
```

该方法共有 4 个参数，这些参数都可省略。各参数的含义如下所述。

- ❑ Before：指定工作表的对象，新建的工作表将置于此工作表之前。
- ❑ After：指定工作表的对象，新建的工作表将置于此工作表之后。
- ❑ Count：要添加的工作表数。默认值为 1。
- ❑ Type：指定工作表类型。可以为 xlWorksheet（工作表）、xlChart（图表）、xlExcel4MacroSheet（Excel 版本 4 宏工作表）或 xlExcel4IntlMacroSheet（Excel 版本 4 国际宏工作表）。默认值为 xlWorksheet。

提示：如果同时省略参数 Before 和 After，则新工作表插入到活动工作表之前。新建的工作表将成为活动工作表。

例如，以下代码可向工作簿中新加工作表：

```
Sub 新增工作表()
    Dim str1 As String
    On Error Resume Next
    str1 = Aplication.InputBox(prompt:="请输入已有工作表名称，" & vbNewLine &
    _
       "新增的工作表将位于该工作表前面。", _
       Title:="输入原工作表名称", Type:=2)
    Worksheets.Add before:=Worksheets(str1)
End Sub
```

执行以上代码，将弹出如图 13-1 所示的对话框，让用户输入原工作表名称，再使用 Add 方法的 Before 参数指定将新增加的工作表放在该工作表之前。

在程序中使用了 On Error 错误捕获语句，用来捕获用户输入的工作表名称不存在时的错误提示。

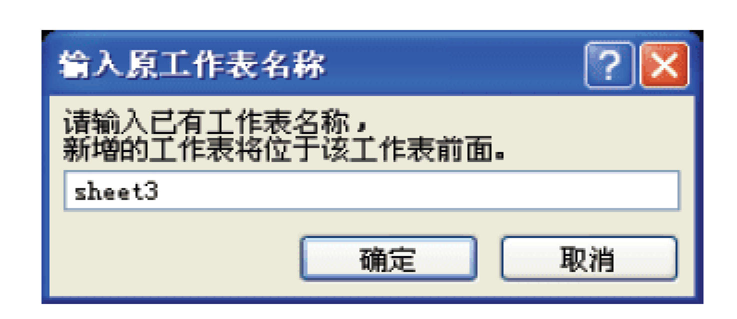


图 13-1　输入原工作表名称

13.2.2　删除工作表

使用 Worksheet 对象的 Delete 方法，可删除指定工作表。

例如，使用以下代码可删除指定的工作表：

```
Sub 删除工作表()
    Dim str1 As String
    On Error GoTo err1
    str1 = Application.InputBox(prompt:="请输入要删除的工作表名称：", _
        Title:="输入工作表名称", Type:=2)
    If str1 = "False" Then Exit Sub

    Application.DisplayAlerts = False             '不显示警告信息
    Worksheets(str1).Delete
    Application.DisplayAlerts = True
    Exit Sub
err1:                                             '错误处理
    MsgBox "不能删除工作表“" & str1 & "”！"
    Application.DisplayAlerts = True
End Sub
```

执行以上代码，将弹出如图 13-2 所示的对话框，用户输入需要删除的工作表名称后，单击【确定】按钮即可删除指定的工作表。

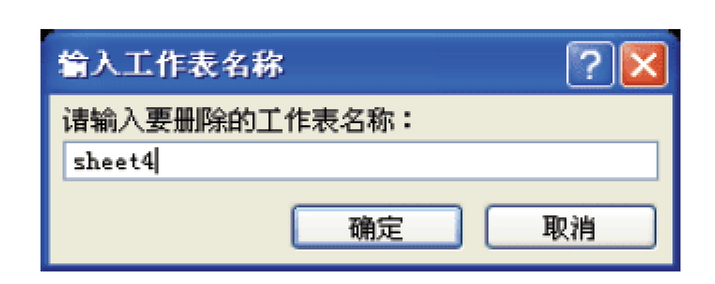


图 13-2　输入工作表名称

当用户输入了不存在的工作表名称，或输入的工作表不允许删除时，将产生错误。在上面的代码中使用 On Error 语句捕获错误，并显示出错误信息。

13.2.3　获取工作表数

Worksheets 集合对象由 Worksheet 对象组成，该集合的 Count 属性返回集合中 Worksheet 对象的数量，代表工作簿中工作表的数量。

使用 Sheets 集合对象的 Count 属性，可返回指定的或活动工作簿中所有工作表的集合。

这里所说的工作表包含 Chart（图表工作表）或 Worksheet 对象。应注意 Worksheets 集合与 Sheets 集合的区别。

例如，以下代码将显示当前工作簿中工作表的数量（不包括图表工作表）：

```
Sub 工作表数量()
    Dim i As Long
    i = Worksheets.Count
    MsgBox "当前工作簿的工作表数为：" & i
End Sub
```

13.2.4 激活工作表

使用 Worksheet 对象的 Activate 方法，将使当前工作表成为活动工作表。使用此方法相当于单击工作表的标签。

例如，以下代码可逐个激活工作簿中的工作表。

```
Sub 逐个激活工作表()
    Dim sh As Worksheet
    For Each sh In Worksheets
        sh.Activate
        MsgBox "激活工作表名称为：" & sh.Name & vbNewLine & _
            "单击【确定】按钮将激活下一工作表！"
    Next
End Sub
```

以上代码使用 For Each 循环对 Worksheets 集合对象中的每个元素重复执行激活语句。运行以上代码将显示如图 13-3 所示的提示对话框。

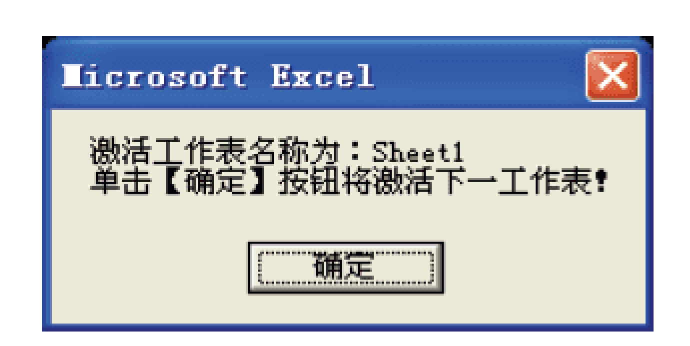


图 13-3　提示对话框

13.2.5 选择工作表

与激活工作表不同，使用 Worksheet 对象的 Select 方法可选择多个工作表。Select 方法的语法格式如下：

```
表达式.Select(Replace)
```

参数 Replace 可省略。该参数如果为 True，则用指定的对象替换当前所选内容。如果为 False，则扩展当前所选内容以包括以前选择的对象和指定的对象。

在选择多个工作表时，如果不将 Replace 参数设置为 False，则执行后一个 Select 方法

时，将取消前一个工作表的选取状态。例如，以下代码最后只有第 3 个工作表处于选中状态：

```
Worksheets(1).Select
Worksheets(3).Select
```

而以下代码将同时选中第 1 个和第 3 个工作表：

```
Worksheets(1).Select
Worksheets(3).Select False
```

使用以下代码也可同时选中多个工作表：

```
Worksheets(Array(Worksheets(1).Name, Worksheets(3).Name)).Select
```

13.2.6　选取前后工作表

在工作簿中按顺序移动工作表时，可以使用 Previous 属性和 Next 属性进行操作。

- Previous 属性：返回代表下一个工作表的 Worksheet 对象。
- Next 属性：返回代表下一个工作表的 Worksheet 对象。

注意：如果当前工作表是第一个工作表，则使用 Previous 属性会出错。如果当前工作表是最后一个工作表，则使用 Next 属性会出错。

可使用以下代码选择前一个工作表：

```
Sub 选择前一个工作表()
    If ActiveSheet.Index <> 1 Then
        ActiveSheet.Previous.Activate
    Else
        MsgBox "已到第一个工作表"
    End If
End Sub
```

以上代码中，ActiveSheet 代表活动工作簿中或指定的窗口、工作簿中的活动工作表（最上面的工作表）。通过 ActiveSheet.Index 属性判断当前工作表是否为第一个工作表，并对用户进行相应的提示。

同样，使用以下代码可选择下一个工作表：

```
Sub 选择后一个工作表()
    If ActiveSheet.Index <> Worksheets.Count Then
        ActiveSheet.Next.Activate
    Else
        MsgBox "已到最后一个工作表"
    End If
End Sub
```

以上代码中，使用 Worksheets 对象的 Count 属性得到工作簿中工作表的数量，用来判断当前工作表是否为最后一个工作表。

13.2.7 工作表保护状态

通过 Worksheet 对象的 ProtectContents 属性，可获取工作表的保护状态。如果工作表内容是受保护的，则为 True。此属性保护单独的单元格。

例如，以下代码可判断当前工作表的保护状态：

```
Sub 工作表保护状态()
    If ActiveSheet.ProtectContents Then
        MsgBox "当前工作表已保护！"
    Else
        MsgBox "当前工作表未保护！"
    End If
End Sub
```

13.2.8 保护工作表

可使用 Worksheet 对象的 Protect 方法保护工作表，使其不能被修改。该方法的语法格式如下：

```
表 达 式 .Protect(Password, DrawingObjects, Contents, Scenarios,
UserInterfaceOnly, AllowFormattingCells, AllowFormattingColumns,
AllowFormattingRows, AllowInsertingColumns, AllowInsertingRows,
AllowInsertingHyperlinks, AllowDeletingColumns, AllowDeletingRows,
AllowSorting, AllowFiltering, AllowUsingPivotTables)
```

该方法有很多参数，这些参数都可以省略，一般只需要设置 Password 密码即可。各参数的含义如下所述。

- Password：该参数为工作表的密码（区分大小写）。如果省略此参数，不用密码就可以取消对工作表的保护。否则，必须指定密码才能取消对工作表的保护。如果忘记了密码，则无法取消对工作表的保护。
- DrawingObjects：该参数如果为 True，则保护工作表中的形状。默认值是 True。
- Contents：该参数如果为 True，则保护内容。对于图表，这样会保护整个图表。对于工作表，这样会保护锁定的单元格。默认值是 True。
- Scenarios：该参数如果为 True，则保护方案。此参数仅对工作表有效。默认值是 True。
- UserInterfaceOnly：该参数如果为 True，则保护用户界面，但不保护宏。如果省略此参数，则既保护宏也保护用户界面。
- AllowFormattingCells：该参数如果为 True，则允许用户为受保护的工作表上的任意单元格设置格式。默认值是 False。
- AllowFormattingColumns：该参数如果为 True，则允许用户为受保护的工作表上的任意列设置格式。默认值是 False。
- AllowFormattingRows：该参数如果为 True，则允许用户为受保护的工作表上的任

意行设置格式。默认值是 False。

- AllowInsertingColumns：该参数如果为 True，则允许用户在受保护的工作表上插入列。默认值是 False。
- AllowInsertingRows：该参数如果为 True，则允许用户在受保护的工作表上插入行。默认值是 False。
- AllowInsertingHyperlinks：该参数如果为 True，则允许用户在受保护的工作表中插入超链接。默认值是 False。
- AllowDeletingColumns：该参数如果为 True，则允许用户在受保护的工作表上删除列，要删除的列中的每个单元格都被解除锁定。默认值是 False。
- AllowDeletingRows：该参数如果为 True，则允许用户在受保护的工作表上删除行，要删除的行中的每个单元格都被解除锁定。默认值是 False。
- AllowSorting：该参数如果为 True，则允许用户在受保护的工作表上进行排序。排序区域中的每个单元格必须是解除锁定的或取消保护的。默认值是 False。
- AllowFiltering：该参数如果为 True，则允许用户在受保护的工作表上设置筛选。用户可以更改筛选条件，但是不能启用或禁用自动筛选功能。用户也可以在已有的自动筛选功能上设置筛选。默认值是 False。
- AllowUsingPivotTables：该参数如果为 True，则允许用户在受保护的工作表上使用数据透视表。默认值是 False。

例如，使用以下代码，可对当前工作簿中的所有工作表进行保护：

```
Sub 保护工作表()
    On Error Resume Next
    Dim ws1 As Worksheet
    Dim str1 As String
    str1 = Application.InputBox(prompt:="请输入保护工作表的密码：", _
        Title:="输入密码", Type:=2)
    For Each ws1 In Worksheets
        ws1.Protect Password:=str1
    Next
    MsgBox "所有工作表保护完成！"
End Sub
```

以上代码使用 For Each 循环处理 Worksheets 集合对象中的每一个 Worksheet 对象，分别对每个工作表设置保护密码。

13.2.9　撤销工作表的保护

对于使用密码保护的工作表，如果需要对保护的工作表进行修改、编辑操作，需要使用 Worksheet 对象的 Unprotect 方法取消工作表的保护。如果工作表或工作簿不是受保护的，则此方法不起作用。Unprotect 方法的语法格式如下：

```
表达式.Unprotect(Password)
```

参数 Password 指定用于解除工作表保护的密码，此密码是区分大小写的。如果工作表

不设密码保护，则省略此参数。如果对工作表省略此参数，而该工作表又设有密码保护，Excel 将提示用户输入密码。

在撤销工作表保护时，如果输入的密码错误，则应给用户提示。可使用 On Error 捕获错误信息。

例如，以下代码要求用户输入密码，对工作簿中的所有工作表撤销保护：

```
Sub 撤销工作表保护()
    On Error GoTo err1
    Dim ws1 As Worksheet
    Dim str1 As String
    str1 = Application.InputBox(prompt:="请输入撤销保护工作表的密码：", _
        Title:="输入密码", Type:=2)
    For Each ws1 In Worksheets
        ws1.Unprotect Password:=str1
    Next
    MsgBox "所有工作表的保护已被撤销！"
    Exit Sub
err1:
    MsgBox "输入的密码错误，不能取撤销对工作表的保护！"
End Sub
```

13.2.10　判断工作表是否存在

当在工作簿中访问不存在的工作表时，系统将产生错误。使用 On Error 语句捕获这种错误信息，就可以判断指定的工作表是否存在。

例如，以下的自定义函数 WorksheetExists 可判断工作表是否存在，该函数的参数为工作表名称，当工作表存在时将返回 True；否则返回 False。

```
Function WorksheetExists(ByVal SheetName As String) As Boolean
    Dim sName As String
    On Error GoTo err1
    sName = Worksheets(SheetName).Name
    WorksheetExists = True
    Exit Function
err1:
    WorksheetExists = False
End Function
```

13.2.11　复制工作表

使用 Worksheet 对象的 Copy 方法，可将工作表复制到工作簿的另一位置。其语法格式如下：

```
表达式.Copy(Before, After)
```

Copy 方法的两个参数都可省略，各参数的含义如下所示。

- ❑ Before：复制的工作表放在该参数指定的工作表之前。如果指定了 After，则不能指定 Before。
- ❑ After：复制的工作表放在该参数指定的工作表之后。如果指定了 Before，则不能指定 After。

注意：如果既不指定 Before 也不指定 After，则 Excel 将新建一个工作簿，其中包含复制的工作表。

例如，以下代码可复制当前工作表。

```
Sub 复制工作表()
    Dim ws1 As Worksheet
    Set ws1 = ActiveSheet
    MsgBox "复制当前工作到前面。"
    ws1.Copy Before:=ws1
    MsgBox "复制当前工作表到后面。"
    ws1.Copy After:=ws1
End Sub
```

以上代码首先将当前工作表（ActiveSheet）赋值给一个对象变量，再使用该对象变量完成工作表的复制操作。

提示：复制的工作表名称为原工作表名称加一个序号。例如，复制工作表 Sheet1 后，得到的工作表为 Sheets1(2)。

13.2.12　隐藏工作表

通过 Worksheet 对象的 Visible 属性，可获取或设置工作表是否可见，可为该属性设置为以下常量之一。

- ❑ xlSheetHidden：隐藏工作表，用户可以通过菜单取消隐藏。
- ❑ xlSheetVeryHidden：隐藏工作表。通过该值隐藏的工作表，用户将不能再通过菜单取消隐藏，只能通过 VBA 代码修改工作表的可见状态。
- ❑ xlSheetVisible：显示工作表。

设置 Visible 属性为 True 或 False，也可显示或隐藏工作表。

例如，使用以下代码可隐藏指定的工作表：

```
Sub 隐藏工作表()
    Dim str1 As String, ws1 As Worksheet
    str1 = Application.InputBox(Prompt:="请输入需要隐藏的工作表：", _
        Title:="隐藏工作表", Default:="Sheet1", Type:=2)
    On Error GoTo err1
    Set ws1 = Worksheets(str1)

    ws1.Visible = xlSheetHidden
```

```
    Exit Sub
err1:
    MsgBox "输入的工作表不存在！"
End Sub
```

以上代码通过错误捕捉语句，处理用户输入工作表不存在的情况。

如果要显示工作表，可将 Worksheet 对象的 Visible 属性设置为 True（或常数 xlSheetVisible）。

提示：用 VBA 代码隐藏的工作表，将不能通过 Excel 界面中的命令显示出来。

13.2.13 移动工作表

使用 Worksheets 集合对象的 Move 方法将工作表移到工作簿中的其他位置。其语法格式如下所述：

```
表达式.Move(Before, After)
```

参数的含义如下所述。

- Before：复制的工作表放在该参数指定的工作表之前。如果指定了 After，则不能指定 Before。
- After：复制的工作表放在该参数指定的工作表之后。如果指定了 Before，则不能指定 After。

如果既不指定 Before 也不指定 After，Excel 将新建一个工作簿，其中包含所移动的工作表。

例如，使用以下代码可将当前工作表移到所有工作表前面：

```
ActiveSheet.Move Before:=Sheets(1)
```

13.2.14 计算工作表打印页数

工作表根据设置的纸张大小划分打印的页，在 Excel 工作表中用虚线划分页。页面可按水平和垂直方向划分，在 VBA 中可通过 HPageBreaks 属性和 VPageBreaks 属性来获取水平和垂直分页符。

- HPageBreaks 属性：返回一个 HPageBreaks 集合，它代表工作表上的水平分页符。
- VPageBreaks 属性：返回一个 VPageBreaks 集合，它代表工作表上的垂直分页符。

每个水平分页符都由一个 HPageBreak 对象代表。如果添加的分页符不和打印区域交叠，则新添加的 HPageBreak 对象将不会出现在打印区域的 HPageBreaks 集合中。如果重新调整打印区域或重新定义打印区域，将会改变集合的内容。

与 HPageBreaks 属性类似，VPageBreaks 集合包含 VPageBreak 对象，每个 VPageBreak 对象代表一个垂直分页符。

使用 VPageBreaks(index)（其中 index 是该分页符的分页符索引号）可返回一个

VPageBreak 对象。

例如，使用以下代码可计算当前工作表的打印页数：

```
Sub 计算页数()
    Dim r As Long, c As Long, p As Long
    Dim ws1 As Worksheet
    Set ws1 = ActiveSheet
    c = ws1.HPageBreaks.Count + 1
    r = ws1.VPageBreaks.Count + 1
    p = r * c
    MsgBox "当前工作表共有" & p & "页。"
End Sub
```

以上代码通过获取 HPageBreaks 集合和 VPageBreaks 集合中包含对象的数量，再将行和列数相乘即可获取工作表中的打印页数。

13.2.15　控制工作表中的图片

在 Excel 工作表中可插入图片。使用 VBA 代码可控制这些插入的图片。

Worksheet 对象的 Shapes 属性可返回一个 Shapes 对象，该对象为工作表上的所有 Shape 对象的集合。每个 Shape 对象都代表绘图层中的一个对象，例如自选图形、任意多边形、OLE 对象或图片。

Shape 对象的 Type 属性可返回或设置一个代表形状类型的 MsoShapeType 枚举值，具体的值如表 13-1 所示。

表 13-1　MsoShapeType枚举类型

名　称	值	描　述	名　称	值	描　述
msoAutoShape	1	自选图形	msoLine	9	线条
msoCallout	2	标注	msoLinkedOLEObject	10	链接 OLE 对象
msoCanvas	20	画布	msoLinkedPicture	11	链接图片
msoChart	3	图	msoMedia	16	媒体
msoComment	4	批注	msoOLEControlObject	12	OLE 控件对象
msoDiagram	21	图表	msoPicture	13	图片
msoEmbeddedOLEObject	7	嵌入的 OLE 对象	msoPlaceholder	14	占位符
msoFormControl	8	窗体控件	msoScriptAnchor	18	脚本定位标记
msoFreeform	5	任意多边形	msoShapeTypeMixed	–2	混和形状类型
msoGroup	6	组合	msoTable	19	表
msoIgxGraphic	24	IGX 图形	msoTextBox	17	文本框
msoInk	22	墨迹	msoTextEffect	15	文本效果
msoInkComment	23	墨迹批注			

例如，使用以下代码可删除当前工作表中的所有图片：

```
Sub 删除图片()
```

```
    Dim p As Shape
    For Each p In ActiveSheet.Shapes
        If p.Type = msoPicture Then p.Delete
    Next
End Sub
```

以上代码循环访问当前工作表中的各个 Shape 对象，如果其类型为 msoPicture（图片），则删除该对象。

13.2.16　处理超链接

Hyperlinks 集合对象代表工作表或区域的超链接的集合。每个超链接都由一个 Hyperlink 对象代表。

1．添加超链接

Hyperlinks 集合对象的 Add 方法可向指定的区域或形状添加超链接。该方法的语法格式如下：

```
表达式.Add(Anchor, Address, SubAddress, ScreenTip, TextToDisplay)
```

各参数的含义如下所述。

- Anchor：为 Object，表示设置超链接的位置。可以是 Range 或 Shape 对象。
- Address：为一个字符串，表示超链接的地址。
- SubAddress：超链接的子地址。
- ScreenTip：当鼠标指针停留在超链接上时所显示的屏幕提示。
- TextToDisplay：要显示的超链接的文本。

例如，使用以下代码可向当前工作表中添加超链接：

```
Sub 添加超链接()
    Dim i As Integer
    With ActiveSheet
        For i = 1 To Worksheets.Count - 1
            .Cells(i + 2, 2).Value = Worksheets(i + 1).Name
            .Hyperlinks.Add anchor:=Cells(i + 2, 2), _
                Address:="", SubAddress:=Cells(i + 2, 2).Value & "!a1", _
                TextToDisplay:=Cells(i + 2, 2).Value
        Next
    End With
End Sub
```

以上代码将各工作表的名称作为超链接添加到当前工作表中。

2．删除超链接

删除工作表中超链接的代码很简单，只需要对当前工作表的 Hyperlinks 集合对象进行遍历，逐个调用 Hyperlink 对象的 Delete 方法即可。

例如，使用以下代码可将当前工作表中的超链接全部删除：

```
Sub 删除超链接()
    Dim h As Hyperlink, hs As Hyperlinks
    Set hs = ActiveSheet.Hyperlinks

    For Each h In hs
        h.Delete
    Next
End Sub
```

13.3　响应用户操作

在 Excel 应用程序中，要控制用户在工作表中的操作，就需要对工作表事件编写代码。工作表事件是开发 Excel 应用程序时使用最多的。

在 VBE 左侧的【工程资源管理器】窗口中双击工作表名称，即可在打开的【代码】窗口中编写工作表事件过程的代码。

13.3.1　禁止选中某个区域

在制作好的工作表中，有时不希望用户选择某些单元格。这时可通过 VBA 代码来禁止用户选择这部分单元格区域。

当工作表上的选定区域发生改变时，将产生 SelectionChange 事件。该事件过程的语法格式如下：

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)

End Sub
```

参数 Target 为新选定的区域。

要禁止用户选择指定的单元格区域，可对 SelectionChage 事件的参数 Target 进行判断，如果其值是不允许用户选择的单元格区域，则强制用户选择其他单元格。

例如，以下代码将禁止用户选择 B1:F3 区域：

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    Dim r As Long, c As Long
    r = Target.Row
    c = Target.Column
    If r <= 3 And c >= 2 And c <= 6 Then [B4].Select
End Sub
```

13.3.2　设置滚动区域

通过在 Worksheet 对象的 SelectionChange 事件中编写代码，可限制用户选择的单元格

区域。但其缺点也显而易见，即用户可单击任何单元格，若该单元格不在允许的区域中，则跳转到指定的单元格。这样就始终有一个单元格跳转的过程。

要设置允许用户使用的单元格区域（其他区域禁止用户选择），更好的方法是使用 Worksheet 对象的 ScrollArea 属性。该属性以 A1 样式的区域引用形式返回或设置允许滚动的区域。用户不能选定滚动区域之外的单元格。

提示： 将 ScrollArea 属性设置为空字符串（""），以允许对整张工作表内所有单元格的选定，即可取消滚动区域的限制。

如果要在用户进入工作表时就限制访问区域，可在工作表的 Activate 事件过程中编写代码。

在激活工作簿、工作表、图表工作表或嵌入式图表时发生 Worksheet 对象的 Activate 事件。

注意： 新建窗口时不发生此事件。

例如，以下代码在工作表的 Activate 事件过程中编写代码，限制工作表的滚动区域为“A1:E65536”，即用户只能选择 A～E 列中的单元格。

```
Private Sub Worksheet_Activate()
    ActiveSheet.ScrollArea = "A1:E65536"
End Sub
```

以上代码设置当前工作表的滚动区域为 A 列至 E 列。因为文件保存为 Excel 2003 格式，所以行数设置为 65 536，若保存为 Excel 2007 格式，应设置为“A1:E1048576”。

13.3.3 禁止输入相同数据

有些情况不允许用户在同一列中输入相同的值。对于数据量少的表格，用户可逐格核对，但是当数据很多时，人工核对容易出错，并且费时费力。这时可使用 VBA 代码进行自动判断。

调用工作表函数 CountIf 可统计指定单元格区域是否有相同的数据。

在 VBA 中，工作表函数 CountIf 以 WorksheetFunction 对象的一个方法形式来使用，用来计算区域中满足给定条件的单元格的个数。其语法格式如下：

```
表达式.CountIf(Arg1, Arg2)
```

各参数的含义如下所述。

- Arg1：用于计算其中满足条件的单元格个数的单元格区域。
- Arg2：用于定义哪些单元格将被计算在内的条件，其形式可以是数字、表达式、单元格引用或文本。例如，条件可以表示为 32、"32"、">32"、"apples"或 B4。

例如，以下代码统计在 B 列中与当前单元格数据相同的单元格数量，如果数量大于 1，则表示有重复的数据。

```
WorksheetFunction.CountIf(Columns(2), Target.Value)
```

在工作表的 Chang 过程中编写以下代码，用来统计当前单元格输入的值是否与第 2 列中已有的值相同。

```
Private Sub Worksheet_Change(ByVal Target As Range)
    Application.EnableEvents = False                '禁止响应事件
    If Target.Column = 2 Then
      If Target.Value <> "" And WorksheetFunction.CountIf(Columns(2), Target.
      Value) > 1 Then
            MsgBox "请不要输入相同的数据！"
            Application.Undo
        End If
    End If
    Application.EnableEvents = True                 '允许响应事件
End Sub
```

一般情况下，如果在 Change 事件过程中编写有代码，当单元格的值进行改变时将执行 Change 事件过程中的代码。如果在 Change 事件过程中又有修改单元格值的情况发生，则将发生事件嵌套调用，导致堆栈溢出，使程序运行出错。

因此，如果需要在 Change 事件过程中修改单元格的值，则先使用以下语句禁止 Excel 响应用户事件：

```
Application.EnableEvents = False
```

在 Change 事件过程结束处，应使用以下语句允许响应用户事件。

```
Application.EnableEvents = True
```

13.3.4　输入连续的数据

在某些工作表中，要求用户在表格中输入数据时数据是连续的，即逐行输入。如果用户跨行输入数据时将导致数据间有空行出现。例如，在第 1～3 行有数据，接着在第 5 行输入数据，则第 4 行就是一个空行。

要达到这样的目的，可以通过 VBA 代码控制用户的操作，这样用户只能将活动单元格移到数据区域的下一行，然后按向下光标键，活动单元格将自动移动到紧靠数据区域的下一个空行。

可在工作表的 SelectionChange 事件过程中编写代码。当工作表上的选定区域发生改变时，将产生 SelectionChange 事件，该事件过程的语法格式如下：

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)

End Sub
```

参数 Target 为新选定的区域。

例如，在工作表 Sheet1 的 SelectionChange 事件中编写代码如下：

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    i = ActiveSheet.Range("A65536").End(xlUp).Row
```

```
    j = Target.Column
    If Target.Row > i Then
        Cells(i + 1, j).Select
    End If
End Sub
```

以上代码首先从当前工作表的最后一行（为了兼容 Excel 2003，此处设置为 65 536 行）向上查找有数据的单元格，并返回其行数。接着判断用户选择的单元格所在行，若超过该行数，则选择下一行对应列作为活动单元格。

13.3.5　增加快捷菜单

要增加工作表的快捷菜单项，需要捕获鼠标的右击操作。使用 Worksheet 对象的 BeforeRightClick 事件可捕获工作表中的右击操作。

右击工作表时发生 BeforeRightClick 事件，此事件先于默认的右击操作。该事件过程的语法格式如下：

```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range, Cancel As
Boolean)

End Sub
```

其中的参数含义如下：

- ❑ Target 表示一个 Range 对象，是双击发生时最靠近鼠标指针的单元格。
- ❑ Cancel 事件发生时为 False。如果事件过程将此参数设为 True，则在完成此过程后，不执行默认的双击操作。

注意：当指针在形状或命令栏（工具栏或菜单栏）上时，右击不触发此事件。

有关创建快捷菜单的命令将在本书第 21 章中进行介绍。

例如，要在工作表 Sheet 中添加快捷菜单，可使用以下代码：

```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range, Cancel As
Boolean)
    For Each mnu1 In Application.CommandBars("cell").Controls
        If mnu1.Tag = "MyMenu" Then mnu1.Delete
    Next
    If Not Application.Intersect(Target, Range("A1:C10")) Is Nothing Then
        With Application.CommandBars("cell").Controls.Add _
            (Type:=msoControlButton, before:=6, temporary:=True)
           .Caption = "测试命令"
           .OnAction = "显示测试信息"
           .Tag = "MyMenu"
        End With
    End If
End Sub
```

以上代码用一段循环语句对单元格快捷菜单项进行逐个判断，如果某个菜单项的标记

（Tag）为 MyMenu（这个标记是在后面添加新菜单时进行设置的，也可设置为需要的名称），则将该菜单项删除，以免在不必要的地方出现该菜单。

接着，使用 Application 对象的 Intersect 方法将右击处的单元格（Target）和预设的区域（“A1:C10”）进行重叠，如果 Target 在“A1:C10”区域内，则在快捷菜单中新增加一个菜单项。该菜单项显示（Caption）为“测试命令”，单击该菜单项后执行的宏（OnAction）为“显示测试信息”（需要另外编写该宏代码），另外设置新增加菜单项的标记（Tag）为 MyMenu，以方便程序访问该菜单项。

当在工作表 Sheet 的区域“A1:C10”中单击鼠标右键时，将显示如图 13-4 所示的快捷菜单。若在其他单元格上单击右键，弹出的快捷菜单中将不会出现【测试命令】菜单项。

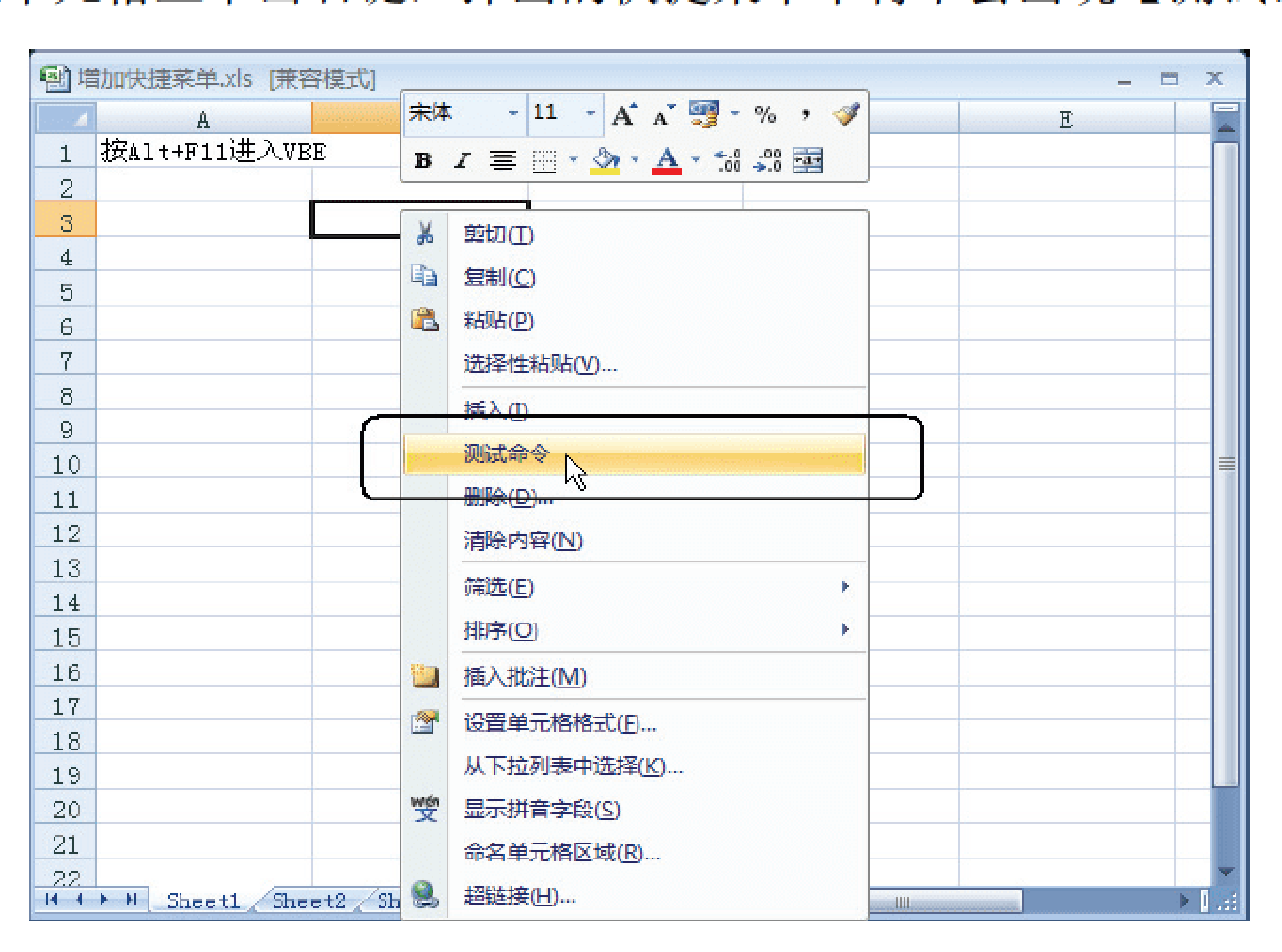


图 13-4　显示增加的快捷菜单项

当用户在快捷菜单中单击【测试命令】菜单后，将调用过程“显示测试信息”完成用户设置的命令。向工程中插入一个模块，编写以下代码响应用户选择新增快捷菜单：

```
Sub 显示测试信息()
    MsgBox "你选择了用户添加的快捷菜单！" & _
        vbCrLf & "本例为测试代码，未编写具体的功能。"
End Sub
```

13.3.6　限制选择其他工作表

在有些应用系统中，可能由于用户的权限，需要限制用户访问工作簿中的其他工作表。当单击其他工作表标签时，将提示用户只能在当前工作表中工作，限制用户选择其他工作表。

这时可在工作表的 Deactivate 事件过程中编写代码。当用户单击其他工作表标签时，当前工作表将发生 Deactivate 事件。在该事件过程中可以编写处理当工作表失去选取焦点时需要执行的操作。其事件过程语法结构如下：

```
Private Sub Worksheet_Deactivate()

End Sub
```

例如，若要限制用户只能在工作表 Sheet1 中操作，可在工作表的 Deactivate 事件过程中编写如下代码：

```
Private Sub Worksheet_Deactivate()
    ActiveSheet.Activate
    MsgBox "您无权操作其他工作表，只能在“Sheet1”工作表中进行操作！", _
        vbCritical + vbOKOnly, "警告"
End Sub
```

在工作表 Sheet1 的 Deactivate 事件过程中编写以上代码后，在 Excel 中单击选择其他工作表标签时将弹出如图 13-5 所示的警告提示框。

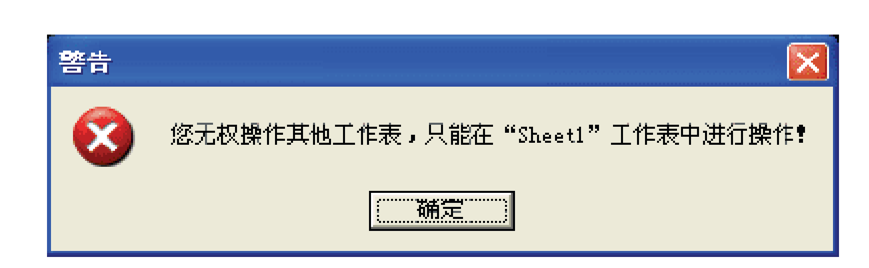


图 13-5　警告信息

13.3.7　隐藏工作表

在某些应用程序中，在用户还未登录进行身份认证之前，不希望查看到其他工作表。这时可只显示作为“主界面”的工作表，将其他工作表隐藏。

用 VBA 代码隐藏的工作表，用户在 Excel 环境中无法通过操作将其显示出来，可起到一定的安全作用。

激活工作簿、工作表、图表工作表或嵌入式图表时将发生 Activate 事件。以下代码将在用户激活名为“主界面”的工作表时，将隐藏除“主界面”之外的所有工作表。

```
Private Sub Worksheet_Activate()
    Dim ws As Worksheet
    For Each ws In Worksheets                '循环隐藏每个工作表
        If ws.Name <> "主界面" Then ws.Visible = False
    Next
End Sub
```

通过 VBA 代码隐藏的工作表必须用 VBA 才能取消其隐藏状态，可使用以下代码将隐藏的工作表全部显示出来：

```
Sub 显示工作表()
    Dim ws As Worksheet
    For Each ws In Worksheets
        ws.Visible = xlSheetVisible
    Next
End Sub
```

13.3.8　突出显示当前位置

要突出显示工作表当前位置，可以在 Excel 中使用【条件格式】设置相应的公式来完成。

当工作表上的选定区域发生改变时将发生 SelectionChange 事件，在该事件过程中编写代码，当发生该事件时就执行该事件过程中的代码。该事件过程的结构如下：

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    事件执行代码
End Sub
```

该事件过程中的参数 Target 为新获得焦点的单元格引用。

例如，在工作表 Sheet1 的 SelectionChange 事件中编写以下代码：

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    Dim i As Integer
    On Error Resume Next
    i = Target.Interior.ColorIndex
    If i < 0 Then
        i = 36
    Else
        i = i + 1
    End If
    If iColor = Target.Font.ColorIndex Then '避免字体颜色与突出色相同
        i = i + 1
    End If
    Cells.Interior.ColorIndex = xlColorIndexNone
    Rows(Target.Row).Interior.ColorIndex = i
    Columns(Target.Column).Interior.ColorIndex = i
End Sub
```

在 SelectionChange 事件过程中编写好以上代码后，在工作表 Sheet1 中选择某一个单元格后，该单元格所在行和列都将以不同的底色显示，以突出单元格的位置。

通过 Range 对象的 Interior 属性返回一个 Interior 对象，该对象代表指定对象的内部。用该对象的属性 ColorIndex 来设置对象的内部颜色。颜色可指定为当前调色板中颜色的索引值，也可设置为以下两个常量之一。

- xlColorIndexAutomatic：自动配色。
- xlColorIndexNone：无色。

第 14 章　使用 Range 对象

在 Excel 应用程序设计中，Range 对象是最常使用的对象之一。在操作 Excel 内的任何区域之前，都需要将其表示为一个 Range 对象，然后使用 Range 对象的方法和属性对其进行操作。

14.1　Range 对象概述

一个 Range 对象代表一个单元格、一行、一列或多个单元格的集合，甚至可以是多个工作表上的一组单元格。本节首先简单介绍 Range 对象的常用属性、方法，在本章后面几节将分别介绍使用 Range 对象操作 Excel 的方法。

14.1.1　Range 对象的常用属性

Range 对象提供了丰富的属性，可以方便地用 VBA 代码从单元格获取或设置值。Range 对象的常用属性包括如下几种。

- Address 属性：返回一个 Range 对象的引用地址。
- Borders 属性：返回一个 Borders 集合，它代表样式或单元格区域（包括定义为条件格式一部分的区域）的边框。
- Characters 属性：返回 Characters 对象，它代表对象文本内某个区域的字符。使用 Characters 对象可为文本字符串内的字符设置格式。
- ColumnWidth 属性：返回或设置指定区域中所有列的列宽。一个列宽单位等于常规样式中一个字符的宽度。对于比例字体，则使用字符 0（零）的宽度。使用 Width 属性可返回以磅为单位的列宽。
- CurrentRegion 属性：返回一个 Range 对象，该对象表示当前区域。当前区域是以空行与空列的组合为边界的区域。
- End 属性：返回一个 Range 对象，该对象代表包含源区域的区域尾端的单元格。等同于按键 Ctrl+↑、Ctrl+↓、Ctrl+←或 Ctrl+→。
- Font 属性：返回一个 Font 对象，它代表指定对象的字体。
- Formula 属性：返回或设置单元格的公式。
- Height 属性：返回或设置单元格的高度（以磅为单位）。
- NumberFormat 属性：返回或设置单元格的格式代码。

- Text 属性：返回或设置单元格中显示的文本。与 Value 属性不同，该属性返回的是单元格中显示出来的内容。而 Value 属性返回的是单元格保存的值。
- Value 属性：返回指定单元格的值。
- Width 属性：返回单元格区域的宽度（以磅为单位）。

14.1.2　Range 对象的常用方法

Range 对象提供了很多方法，可以方便地用 VBA 代码控制单元格。下面列出常用方法及其用法。

- Activate 方法：激活单个单元格，该单元格必须处于当前选定区域内。
- AddComment 方法：为区域添加批注。
- AutoFit 方法：更改区域中的列宽或行高以达到最佳匹配。
- Clear 方法：清除 Range 对象中的内容。
- ClearComments 方法：清除指定区域的所有单元格批注。
- ClearContents 方法：清除区域中的公式。
- ClearFormats 方法：清除对象的格式设置。
- Copy 方法：将单元格区域复制到指定的区域或剪贴板中。
- Cut 方法：将对象剪切到剪贴板，或者将其粘贴到指定的目的地。
- Delete 方法：删除指定的 Range 对象。通过参数为 xlShiftToLeft（单元格向左移动）或 xlShiftUp（单元格向上移动）指定如何调整单元格以替换删除的单元格。
- Find 方法：在区域中查找特定信息。
- Insert 方法：在工作表或宏表中插入一个单元格或单元格区域，其他单元格相应移位以腾出空间。
- Merge 方法：由指定的 Range 对象创建合并单元格。
- Select 方法：选择单元格对象。
- UnMerge 方法：将合并区域分解为独立的单元格。

14.2　引用 Range 对象

在 Excel 中使用 VBA 编程时，需要频繁地引用单元格区域，然后再使用相应的属性和方法对区域进行操作。单元格区域包括以下几种情况。

- 单个的单元格；
- 多个单元格（连续或非连续）组成的区域；
- 整行或整列等。

要用 VBA 代码控制 Range 对象，必须首先获取 Range 对象的引用。在 VBA 中，可通过多种方式获取 Range 对象的引用，本节介绍引用 Range 对象的常用方法。

14.2.1　使用 A1 样式引用单元格

使用 A1 样式引用单元格是 Excel 中最常用的方法。

通过工作表（Worksheet）对象或 Range 对象的 Range 属性返回一个 Range 对象，它代表一个单元格或单元格区域。Range 属性返回一个单元格或单元格区域，对区域的引用如果使用 A1 样式，需将引用字符串包含在引号中。另外还可以使用以下方式引用单元格：

```
[D5]：引用单元格“D5”；
ActiveCell：当前单元格。
```

常见的表示方法如下：

```
Worksheets("sheet1 ").Range("A1")
Worksheets("sheet1 ").Range("A1:B5")
Worksheets("sheet1 ").Range("C5:D9,G9:H16,B14:D18")
Range("MyRange, YourRange, HisRange")
```

其中 MyRange，YourRange，HisRange 为单元格区域名称。

14.2.2　使用索引号引用单元格

Excel 的工作表是一个二维结构，由行和列构成。通过使用行列索引号，可用 Cells 属性引用单个单元格。该属性返回代表单个单元格的 Range 对象。例如，Cells(6,1)返回对工作表中单元格 A6 的引用。

使用 Cells 属性引用单元格时，因为可用变量替代行列索引号，所以 Cells 属性非常适合于在单元格区域中循环。

另外，使用 Cells 属性还可按以下方式引用单元格区域：

```
Cells(5, "D")：表示第 5 行 D 列（即单元格“D5”）
```

如果使用 Cells 属性时不指定行列索引号，该方法将返回代表工作表上所有单元格的 Range 对象。例如，以下代码将清除活动工作簿中 Sheet1 上的所有单元格的内容。

```
Worksheets("Sheet1").Cells.ClearContents
```

例如，以下代码将在当前工作表的单元格区域 A1:E10 中，按顺序在单元格中填充序号。

```
Sub 单元格序号()
    Dim i As Long, j As Long
    For i = 1 To 10
        For j = 1 To 5
            Cells(i, j).Value = (i - 1) * 5 + j
        Next
    Next
```

```
End Sub
```

运行以上代码，当前工作表各单元格的序号如图 14-1 所示。

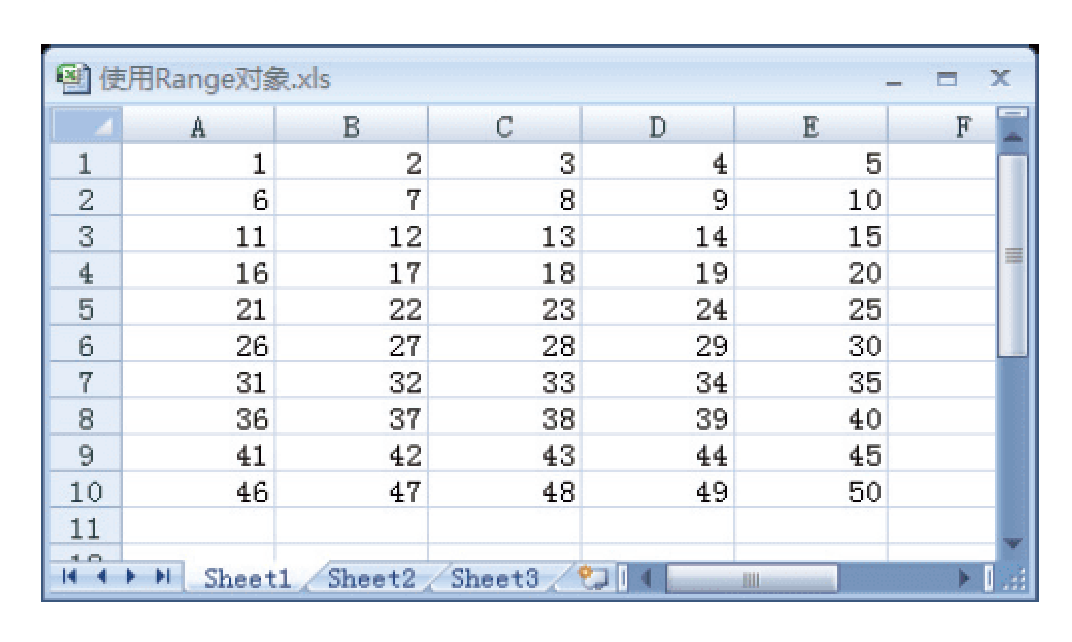


图 14-1　单元格序号

还可以使用以下表示方法来引用 Range 对象：

```
Worksheets("sheet1 ").Cells(1,1)      '第 1 行第 1 列（即 A1 单元格）
Range(Cells(1, 1), Cells(5, 3))       'A1:C5 单元格区域
Worksheets("sheet1 ").Cells           '工作表 sheet1 的所有单元格
```

14.2.3　偏移引用单元格

使用 Range 对象的 Offset 属性返回一个 Range 对象，它代表位于指定单元格区域的一定的偏移量位置上的区域。其使用格式为：

```
Offset(RowOffset, ColumnOffset)
```

两个参数的含义如下：

- ❑ RowOffset 为行偏移量，区域偏移的行数可为正数、负数或 0（零）。正数表示向下偏移，负数表示向上偏移。默认值是 0。
- ❑ ColumnOffset 为列偏移量，区域偏移的列数可为正数、负数或 0（零）。正数表示向右偏移，负数表示向左偏移。默认值是 0。

例如，以下代码返回从指定单元格偏移一定位置的 Range 对象。

```
Range("D3").Offset(, -1)                                  表示单元格 C3
Range("A1").Offset(2, 2)                                  表示单元格 C3
ActiveCell.Offset(rowOffset:=3, columnOffset:=3).Activate 当前单元格向右 3
                                                          列向下 3 行
```

14.2.4　引用行或列

在 VBA 代码中，可以使用多种方法引用工作表中的整行或整列。在使用 A1 样式引用 Range 对象时，只指定行号或列标即可引用行或列。以下的代码即可分别引用列或行：

```
Range("A:C").Select 表示选择 A 列至 C 列
Range("1:1").Select 表示选择第 1 行
Range("1:3").Select 表示选择第 1 行至第 3 行
Range("A:A").Select 表示选择 A 列
```

另外，对指定的 Range 对象，使用以下两个属性也可分别引用列或行。

- EntireColumn 属性：返回一个 Range 对象，该对象表示包含指定区域的整列（或多列）。
- EntireRow 属性：返回一个 Range 对象，该对象表示包含指定区域的整行（或多行）。

例如，以下代码将引用指定的列或行。

```
Range("A3").EntireColumn    表示 C 列
Range("A3").EntireRow    表示第 1 列
```

14.2.5 查找数据区域边界

在 Excel 中，按组合键 Ctrl+光标键可在数据区域中查找到数据区域的边界。在 VBA 中可通过使用 Range 对象的 End 属性来进行相同的操作。

使用 Range 对象的 End 属性将返回一个 Range 对象，该对象代表包含源区域的区域尾端的单元格。其语法格式如下：

```
表达式.End(Direction)
```

参数 Direction 指定移动的方向，可为以下常量之一。

- xlDown：向下移动；
- xlToLeft：向左移动；
- xlToRight：向右移动；
- xlUp：向上移动。

例如，下面的代码选定指定单元格数据区域边缘的单元格。

```
Range("B4").End(xlUp).Select        ' 单元格 B4 中 B 列顶端单元格
Range("C5").End(xlUp).Select        ' 单元格 C5 中 C 列顶端单元格
Range("C5").End(xlToRight).Select   ' 单元格 C5 中第 5 行尾端单元格
Range("C5").End(xlToLeft).Select    ' 单元格 C5 中第 5 行左侧单元格
Range("C5").End(xlDown).Select      ' 单元格 C5 中 C 列底端单元格
```

使用 Range 对象的 End 属性，在 VBA 中可进行很多选择区域的设置。例如，以下代码可选择指定行中连续的数据区域。

```
Sub 选择连续数据区域()
    Dim rng1 As Range
    Dim i As Long , col As Long
    i = Application.InputBox(prompt:="请输入要选择数据的行: ", Type:=1)
    col = Range("A" & i).End(xlToRight).Column
```

```
    Set rng1 = Range(Cells(i, 1), Cells(i, col))
    rng1.Select
    Set rng1 = Nothing
End Sub
```

以上代码要求用户输入需要选中数据的行数，再使用 End 属性找到该列数据区域的右侧单元格，最后选中连续的数据区域。

以上代码中，设置 Application 对象的 InputBox 方法的 Type 参数为 1，表示接收用户输入的数据为数值型。

代码 Range("A" & i).End(xlToRight).Column 获取最右侧的列数。

14.2.6　引用当前区域

所谓当前区域，是指以空行与空列的组合为边界的区域（即如果工作表中数据区域中有一个空行或空列，则将为两个区域）。

使用 Range 对象的 CurrentRegion 属性，可返回一个表示当前区域的 Range 对象。其语法格式如下：

```
表达式.CurrentRegion
```

该属性对于许多自动展开选择区域，以包括整个当前区域的操作很有用。

注意：该属性不能用于被保护的工作表。

例如，以下代码获取一个单元格的引用，再以该单元格为基础，使用 CurrentRegion 属性取得该单元格所在的当前区域。

```
Dim rng1 As Range
Set rng1 = Range("A1")
Set rng1 = rng1.CurrentRegion
```

14.2.7　获取已使用区域

如果想知道当前工作表中所有已使用的单元格区域的大小，或者想引用当前工作表中已使用的区域，可以使用 Worksheet 对象的 UsedRange 属性。

UsedRange 属性返回指定工作表中已使用区域的 Range 对象，即返回工作表中已使用的单元格区域。因此，该属性也可以用于选取单元格区域。

UsedRange 属性与 CurrentRegion 属性的区别主要有以下两点：

- UsedRange 属性是 Worksheet 对象的一个属性，返回的是指定工作表中所有已使用的单元格区域，无论各单元格之间是否有空行或空列隔开。
- CurrentRegion 属性是 Range 对象的一个属性，返回的是一个由空行空列围起来的区域，空行空列之外的单元格不被包含在内。

例如，在如图 14-2 所示的工作表数据中，第 6 行为空。

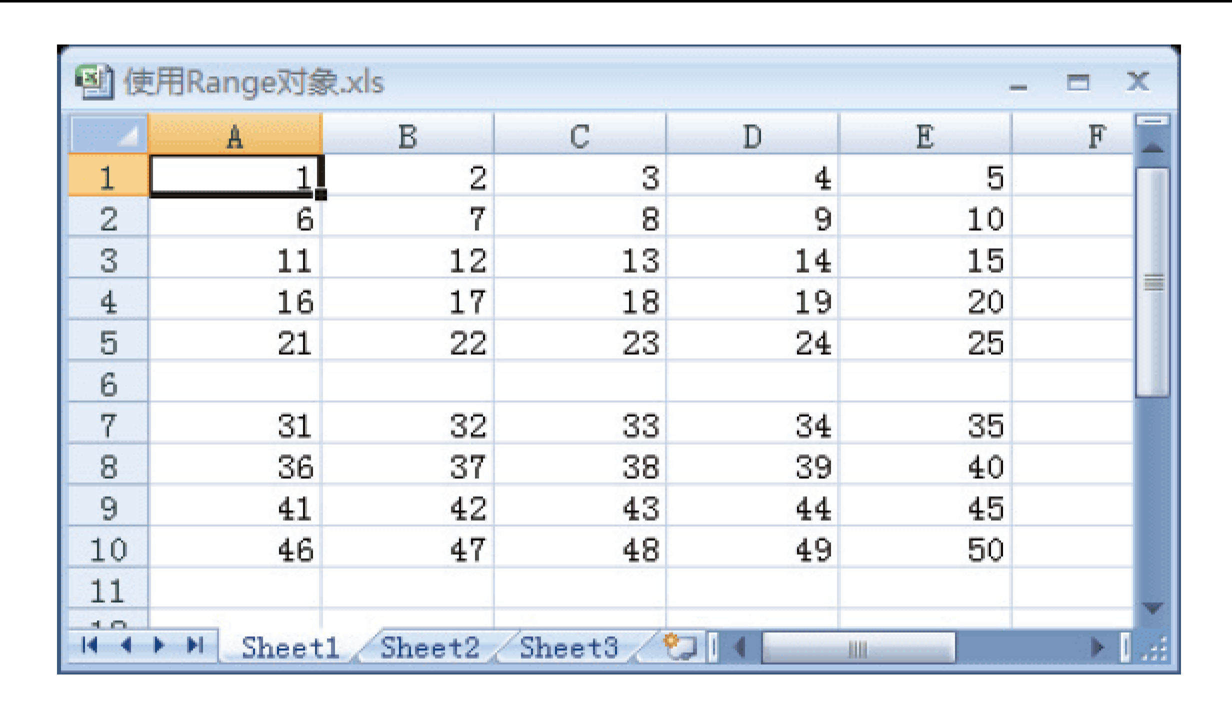


图 14-2　工作表数据

使用以下代码选中活动工作表中的使用区域 A1:E10。

```
ActiveSheet.UsedRange.Select
```

而使用以下语句选中的区域为“A1:E5”，因为第 6 行为空行，所以后面的单元格将不被包含在内。

```
Range("A1").CurrentRegion.Select
```

14.2.8　获取重叠区域引用

使用 Application 对象的 Intersect 方法，可以返回一个 Range 对象，该对象包含两个或多个区域重叠的矩形区域。其语法格式如下：

```
表达式.Intersect(Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9, Arg10,
Arg11, Arg12, Arg13, Arg14, Arg15, Arg16, Arg17, Arg18, Arg19, Arg20, Arg21,
Arg22, Arg23, Arg24, Arg25, Arg26, Arg27, Arg28, Arg29, Arg30)
```

提示：使用 Intersect 方法时，至少需要 2 个 Range 区域，最多 30 个 Range 区域进行交叉计算。

例如，以下代码将返回两个单元格区域的重叠区域：

```
Sub 重叠区域()
    Dim rng1 As Range, rng2 As Range, rng3 As Range
    Set rng2 = Range("A1:E10")
    Set rng3 = Range("C1:C10")
    Set rng1 = Application.Intersect(rng2, rng3)
    MsgBox rng1.Address
End Sub
```

使用 Application 对象的 Intersect 方法时，如果多个区域没有重叠的部分，则将返回 Nothing。在 VBA 程序中可使用判断语句对其进行判断，以免引用单元格时出错。

14.2.9　获取合并区域引用

使用 Application 对象的 Union 方法可将多个单元格区域组合到一个 Range 对象中。Application.Union 方法的语法格式如下：

```
表达式.Union(Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9, Arg10,
Arg11, Arg12, Arg13, Arg14, Arg15, Arg16, Arg17, Arg18, Arg19, Arg20, Arg21,
Arg22, Arg23, Arg24, Arg25, Arg26, Arg27, Arg28, Arg29, Arg30)
```

使用 Union 方法时至少需要 2 个单元格区域，但最多可合并 30 个单元格区域。例如：

```
Union(Range("A1:C3"), Range("E1:F5"))
```

以上语句与下面的代码引用相同的单元格区域：

```
Range("A1:C3, E1:F5 ")
```

将 Range 对象的 End 属性和 Union 方法结合使用，可选择工作表中的不连续数据区域。例如，以下代码将选择 A 列和 E 列中的所有数据单元格：

```
Sub 选择不连续数据区域()
    Dim rng1 As Range, rng2 As Range
    Set rng1 = Range("A1", Range("A1").End(xlDown))
    Set rng2 = Range("E1", Range("E1").End(xlDown))
    Union(rng1, rng2).Select
    Set rng1 = Nothing
    Set rng2 = Nothing
End Sub
```

以上代码使用 End 属性选择同一行含有数据的区域，再使用 Union 方法将多个区域合并在一起。

14.2.10　获取指定类型的单元格

在 Excel 工作表中，使用条件格式可以帮助用户直观地查看和分析数据、发现关键问题以及识别模式和趋势。

在 VBA 中，使用 Range 对象的 SpecialCells 方法可返回一个 Range 对象，该对象代表与指定类型和值匹配的所有单元格。该方法的语法格式如下：

```
表达式.SpecialCells(Type, Value)
```

参数 Type 为一个常量，指定要包含的单元格，常用的常量包括如下几种。

- xlCellTypeAllFormatConditions：含有条件格式的单元格。
- xlCellTypeAllValidation：含有验证条件的单元格。

- xlCellTypeBlanks：空单元格。
- xlCellTypeComments：含有注释的单元格。
- xlCellTypeConstants：含有常量的单元格。
- xlCellTypeFormulas：含有公式的单元格。
- xlCellTypeLastCell：已用区域中的最后一个单元格。
- xlCellTypeSameFormatConditions：含有相同格式的单元格。
- xlCellTypeSameValidation：含有相同验证条件的单元格。
- xlCellTypeVisible：所有可见的单元格。

参数 Value 可省略。如果参数 Type 为 xlCellTypeConstants 或 xlCellTypeFormulas，则该参数可用于确定结果中应包含哪几类单元格。将这些值相加可使此方法返回多种类型的单元格。默认情况下，将选择所有常量或公式，无论类型如何。该参数可为下述的常量值。

- xlErrors：错误值；
- xlLogical：逻辑值；
- xlNumbers：数值；
- xlTextValues：文本。

例如，以下代码选定工作表 Sheet1 中已用区域的最后一个单元格：

```
Sheet1.Cells.SpecialCells(xlCellTypeLastCell).Activate
```

以下代码选择工作表 Sheet1 中设置了条件格式的单元格：

```
Sheet1.Cells.SpecialCells(xlCellTypeAllFormatConditions)
```

使用以下代码，可选中不含公式的单元格：

```
Sub 不含公式的单元格()
    Dim rng As Range, rng1 As Range, rng2 As Range
    Set rng = ActiveSheet.UsedRange                '当前使用单元格区域
    rng.SpecialCells(xlCellTypeFormulas).Select '选择公式单元格
    For Each rng1 In rng
        If Application.Intersect(rng1, Selection) Is Nothing Then
                                                   '单元格在公式区域
            If rng2 Is Nothing Then
                Set rng2 = rng1
            Else
                Set rng2 = Union(rng2, rng1)  '合并区域
            End If
        End If
    Next
    rng2.Select                                   '选择区域
End Sub
```

以上代码首先获取当前工作表的使用区域，接着选中含有公式的单元格区域，再通过循环将每个单元格与选中的公式单元格区域进行交集运算，判断指定的单元格是否在公式

单元格选集中，若不在公式单元格选集中，则将其进行合并。最后选中合并的单元格区域。

14.2.11　引用合并区域的子区域

当在 Excel 中选中多个单元格区域时，某些操作不能在选定区域内的多个子区域上同时执行，必须在选定区域内的单个子区域上循环，对每个单独的子区域分别执行该操作。

在 Excel 中，选定多个区域后将生成 Areas 集合，Areas 集合内的各个成员是 Range 对象。在 Areas 集合中，选定区域内每个离散的连续单元格区域都有一个 Range 对象。如果选定区域内只有一个子区域，则 Areas 集合包含一个与该选定区域对应的 Range 对象。

例如，以下代码演示了引用子区域的方法：

```
Sub 引用子区域()
    Dim rng1 As Range
    Dim rng2 As Range
    Set rng1 = Range("A1:C3, E1:F5, A8:C9")
    rng1.Select
    For i = 1 To rng1.Areas.Count
        Set rng2 = rng1.Areas(i)
        str1 = "子区域" & i & vbCr & vbCr
        str1 = str1 & "行数: " & rng2.Rows.Count & vbCr
        str1 = str1 & "列数: " & rng2.Columns.Count
        MsgBox str1
    Next
    Set rng1 = Nothing
    Set rng2 = Nothing
End Sub
```

以上代码的 Areas 集合包括 3 个子区域，通过循环分别显示每个子区域所点的行数和列数。执行的结果如图 14-3 所示。

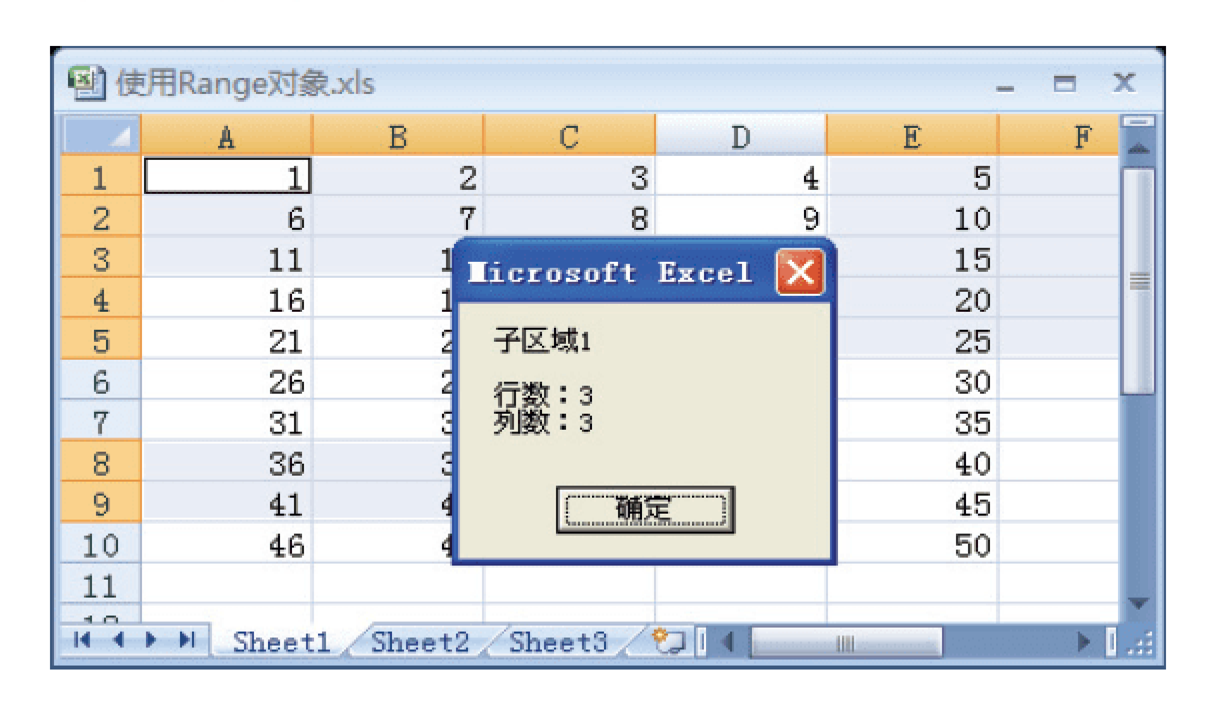


图 14-3　引用合并区域的子区域

14.2.12　引用区域内的单个单元格

要获取选中区域中的某个单元格的值，可以使用 Range 对象的 Item 属性来访问。Item

属性返回一个 Range 对象，它代表对指定区域某一偏移量处的区域。其语法格式如下：

```
表达式.Item(RowIndex, ColumnIndex)
```

各参数的含义如下：

- ❑ RowIndex 是必选的，表示要访问的单元格的索引号，顺序为从左到右，然后往下。
- ❑ ColumnIndex 可省略，用来指明要访问的单元格所在的列号的数字或字符串，用数字 1 或字符“A”表示区域中的第 1 列。

例如：

```
Range("A1:E10").Item(4,3)
```

表示单元格 C4，这个单元格处于以指定区域中左上角单元格 A1（即区域中第 1 行第 1 列的单元格）为起点的第 4 行第 3 列。因为 Item 属性为默认属性，因此也可以简写为：

```
Range("A1:E10")(4,3)
```

当省略 ColumnIndex 参数时，计数方式为从左向右，即在区域中的第 1 行开始从左向右计数，第 1 行结束后，从第二行开始从左到右接着计数，依此类推。例如：

```
Range("A1:B2")(1)    代表单元格 A1
Range("A1:B2")(2)    代表单元格 B1
Range("A1:B2")(3)    代表单元格 A2
Range("A1:B2")(4)    代表单元格 B2
```

技巧： Item 属性的值即可为正数，也可为负数。其序号值指定的单元格不一定包含在引用的区域内。

14.2.13 扩展单元格区域

使用 Range 对象的 Resize 属性，可以调整指定单元格区域的大小，并返回一个 Range 对象，该对象代表调整后的区域。其语法格式如下：

```
表达式.Resize(RowSize, ColumnSize)
```

其两个参数的含义如下：

- ❑ RowSize 为新区域中的行数。如果省略该参数，则该区域中的行数保持不变。
- ❑ ColumnSize 为新区域中的列数。如果省略该参数，则该区域中的列数保持不变。

例如，下列代码将激活 Sheet1 工作表活动单元格向右偏移 3 列、向下偏移 3 行处的单元格。

```
Worksheets("Sheet1").Activate
ActiveCell.Offset(rowOffset:=3, columnOffset:=3).Activate
```

14.3　获取单元格信息

通过不同的方式获取对 Excel 工作表单元格区域的引用后，在 VBA 中引用这些对象即可获取单元格区域的各种信息。

14.3.1　获取单元格地址

使用 Range 对象的 Address 属性可返回代表宏语言的区域引用。其语法格式如下：

```
表达式.Address(RowAbsolute, ColumnAbsolute, ReferenceStyle, External, RelativeTo)
```

其中的各参数都可省略，各参数的含义分别如下所述。

- RowAbsolute：如果为 True，则以绝对引用返回引用的行部分。默认值为 True。
- ColumnAbsolute：如果为 True，则以绝对引用返回引用的列部分。默认值为 True。
- ReferenceStyle：设置为常量 xlA1 返回 A1 样式的引用，常量 xlR1C1 返回 R1C1 样式的引用。
- External：如果为 True，则返回外部引用。如果为 False，则返回本地引用。默认值为 False。
- RelativeTo：如果 RowAbsolute 和 ColumnAbsolute 为 False，并且 ReferenceStyle 为 xlR1C1，则必须包括相对引用的起始点。此参数是定义起始点的 Range 对象。

如果引用包含多个单元格，RowAbsolute 和 ColumnAbsolute 将应用于所有的行和列。

例如，以下代码将返回活动单元格各地址的表示形式：

```
Sub 当前单元格地址()
    Dim rng1 As Range
    Dim str1 As String, strTitle As String
    Set rng1 = ActiveCell
    strTitle = "当前单元格地址"
    str1 = "绝对地址：" & rng1.Address & vbCrLf
    str1 = str1 & "行的绝对地址：" & rng1.Address(RowAbsolute:=False) & vbCrLf
    str1 = str1 & "列的绝对地址：" & rng1.Address(ColumnAbsolute:=False) &
    vbCrLf
    str1 = str1 & "以 R1C1 形式显示：" & rng1.Address(ReferenceStyle:=xlR1C1)
    & vbCrLf
    str1 = str1 & "相对地址：" & rng1.Address(False, False)
    MsgBox prompt:=str1, Title:=strTitle
End Sub
```

在 Excel 工作表中选择单元格 A1。执行以上代码，将显示如图 14-4 所示的对话框，在该对话框中将显示单元格地址的各种表示形式。



图 14-4 【当前单元格地址】对话框

14.3.2 获取区域信息

在 Excel 工作表中，当选择区域后，可使用 Range 对象的多个属性值来获取区域中的相应信息，例如，获取区域的单元格数量、行数、列数等。可使用以下属性来获得这些信息。

- ActiveCell 对象：代表活动窗口（当前工作表）的活动单元格。
- Rows 属性：代表指定单元格区域中的行，通过其 Count 属性可取得行的数量。
- Columns 属性：代表指定单元格区域中的列，通过其 Count 属性可取得列的数量。
- ListHeaderRows 属性：代表指定区域的标题行数。
- Cells 属性：代表指定单元格区域中的单元格，通过其 Count 属性取得单元格的数量。

例如，以下代码将显示当前区域的相关信息：

```
Sub 当前区域信息()
    Dim rng1 As Range, str1 As String
    Set rng1 = ActiveCell.CurrentRegion
    str1 = "当前区域信息： " & vbNewLine
    str1 = str1 & "单元格数量： " & rng1.Cells.Count & vbNewLine
    str1 = str1 & "行数： " & rng1.Rows.Count & vbNewLine
    str1 = str1 & "列数： " & rng1.Columns.Count & vbNewLine
    str1 = str1 & "表头行数： " & rng1.ListHeaderRows
    MsgBox str1, vbOKOnly, "当前区域信息"
    Set rng1 = Nothing
End Sub
```

执行以上代码，将显示如图 14-5 所示的信息。

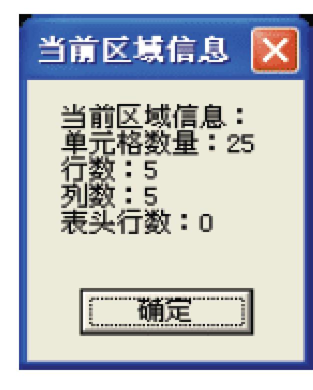


图 14-5 【当前区域信息】对话框

14.3.3 统计区域中公式数量

使用 Range 对象的 HasFormula 属性，可对指定区域是否包含公式进行判断。如果区域

中所有单元格均包含公式，则该属性值为 True；如果所有单元格均不包含公式，则该属性值为 False；其他情况下为 Null。

例如，使用以下代码可显示当前区域公式的数量：

```
Sub 当前区域公式数量()
    Dim c1 As Range, rng1 As Range
    Dim i As Integer
    Set rng1 = ActiveCell.CurrentRegion
    For Each c1 In rng1.Cells
        If c1.HasFormula Then
            i = i + 1
        End If
    Next
    If i > 0 Then
        MsgBox "当前单元格所在区域共有 " & i & " 个公式。"
    Else
        MsgBox "当前单元格所在区域没有公式。"
    End If
End Sub
```

以上代码使用 For Each 循环对当前区域的每个单元格进行单独判断，如果单元格中包括有公式，则累加公式计数器。

14.3.4　追踪公式单元格

在 Excel 中定义公式时，使用了两个概念，即引用单元格和从属单元格。

- 引用单元格：是被其他单元格中的公式引用的单元格。例如，如果单元格 D10 包含公式“=B5”，那么单元格 B5 就是单元格 D10 的引用单元格。
- 从属单元格：该单元格中包含引用其他单元格的公式。例如，如果单元格 D10 包含公式“=B5”，那么单元格 D10 就是单元格 B5 的从属单元格。

为了帮助用户检查公式，可以使用【追踪引用单元格】和【追踪从属单元格】命令以图形方式显示或追踪这些单元格与包含追踪箭头的公式之间的关系。在 VBA 中，可以使用 Range 对象的以下几个属性来追踪这些单元格。

- DirectPrecedents 属性：该属性返回一个 Range 对象，该对象表示包含一个单元格的所有直接引用单元格的区域。如果有多个引用单元格，则可能有多个选择（多个 Range 对象）。
- Precedents 属性：该属性返回一个 Range 对象，该对象表示公式单元格的所有引用单元格。如果有多个引用单元格，则可以是一个多重选择（Range 对象的并集）。
- ShowPrecedents 方法：该方法绘制从指定区域指向直接引用单元格的追踪箭头。
- ShowDependents 方法：该方法绘制从指定区域指向直接从属单元格的追踪箭头。

例如，使用以下代码可以显示引用单元格：

```
Sub 追踪引用单元格()
```

```
    ActiveCell.ShowPrecedents
End Sub
```

而使用以下代码可以显示从属单元格：

```
Sub 追踪从属单元格()
    ActiveCell.ShowDependents
End Sub
```

14.3.5 按颜色统计单元格数量

使用 Range 对象的 Interior 属性返回一个 Interior 对象，该对象代表指定对象的内部。通过 Interior 对象的 ColorIndex 属性可获取单元格内部的颜色。

单元格内部颜色可指定为当前调色板中颜色的索引值（1～56），也可指定为下列 XlColorIndex 常量之一。

- xlColorIndexAutomatic：表示自动配色；
- xlColorIndexNone：表示无色。

提示：也可使用 Interior 对象的 Color 属性来设置内部颜色，该颜色值由 RGB 函数产生。

例如，使用以下代码可统计当前工作表使用区域中，各填充颜色单元格的数量：

```
Sub 按颜色统计单元格()
    Dim rng As Range, rng1 As Range
    Dim i As Integer, Arr(1 to 56) As Integer, k As Integer
    Set rng = ActiveSheet.UsedRange              '获取使用区域
    For Each rng1 In rng                         '循环处理区域中的每个单元格
        k = rng1.Interior.ColorIndex             '获取填充色
        If k <> xlColorIndexNone Then            '具有底色
            Arr(k) = Arr(k) + 1                  '对应颜色数组中进行累加
        End If
    Next
    i = 8                                        '统计单元格显示的位置
    For k = 1 To 56
        If Arr(k) <> 0 Then
            Cells(i, 1).Interior.ColorIndex = k
            Cells(i, 2) = Arr(k)
            i = i + 1
        End If
    Next
End Sub
```

以上代码首先声明一个数组，用来保存每种颜色单元格的数量，接着使用 For Each 循环对每个单元格进行统计，在统计时按单元格内部对象 Interior 的 ColorIndex 值判断单元格内部颜色。最后使用一个循环分别显示出不同的颜色，并在单元格右侧显示该种颜色单元格的数量。

使用 Interior 对象的属性还可获取选定区域的很多相关属性，例如下面的两个属性。

- Pattern 属性：返回或设置一个包含 xlPattern 常量的值，代表内部图案。
- PatternColorIndex 属性：返回或设置内部图案的颜色，表示为当前调色板中的颜色编号或常量 xlColorIndexAutomatic、xlColorIndexNone 之一。

14.4 操 作 行 列

在工作表的操作中，对行列的操作是最常用的操作之一，本节介绍操作行和列方面的内容，包括插入行（列）、隐藏行（列）、设置行高（列宽）等内容。

14.4.1 插入行

插入行的操作可通过 Range 对象的 Insert 方法来完成。

```
表达式.Insert(Shift, CopyOrigin)
```

Insert 方法的两个参数都可以省略，各参数的含义如下所示。

- Shift：指定单元格的调整方式，设置其他单元格相应移位以腾出空间。可以为常量 xlShiftToRight（向右移动单元格）或 xlShiftDown（向下移动单元格）。如果省略此参数，Excel 将根据区域的形状确定调整方式。
- CopyOrigin：复制的起点。

如果要插入行，首先通过工作表的 Rows 属性返回一个 Range 对象，再通过 Range 对象的 Insert 方法插入单元格或单元格区域。具体的代码如下：

```
Sub 插入行()
    Dim r As Long
    r = Selection.Row
    ActiveSheet.Rows(r).Insert
End Sub
```

以上代码省略 Insert 方法的参数，插入行后当前单元格所在行将向下移动。

14.4.2 插入列

与插入行类似，首先通过工作表的 Columns 属性返回一个 Range 对象，再通过 Range 对象的 Insert 方法插入单元格或单元格区域。该方法的语法格式如下：

```
表达式.Insert(Shift, CopyOrigin)
```

Insert 方法的两个参数都可以省略，参数含义如下所述。

- Shift：指定单元格的调整方式，设置其他单元格相应移位以腾出空间。可以为常量 xlShiftToRight（向右移动单元格）或 xlShiftDown（向下移动单元格）。如果省

略此参数，Excel 将根据区域的形状确定调整方式。

- CopyOrigin：复制的起点。

例如，使用以下代码可在选择的单元格中插入一列：

```
Sub 插入列()
    Dim c As Long
    c = Selection.Column
    ActiveSheet.Columns(c).Insert
End Sub
```

14.4.3　删除行

使用 Range 对象的 Delete 方法可删除指定的行。

例如，使用以下代码可删除第 1 行中单元格为空的行：

```
Sub 删除空行()
    Dim rng As Range
    Set rng = Columns(1).SpecialCells(xlCellTypeBlanks)
    rng.EntireRow.Delete
End Sub
```

以上代码使用了 SpecialCells 方法，使用该方法选择满足条件的单元格，其语法格式如下：

```
表达式.SpecialCells(Type, Value)
```

参数的含义分别如下所述。

- Type：设置要包含的单元格。
- Value：如果 Type 为 xlCellTypeConstants 或 xlCellTypeFormulas，则该参数可用于确定结果中应包含哪几类单元格。将这些值相加可使此方法返回多种类型的单元格。默认情况下，将选择所有常量或公式，无论类型如何。

14.4.4　隐藏行

设置 Rows 的 Hidden 属性为 True，可隐藏指定的行。

例如，下面的代码首先通过工作表的 Rows 属性返回一个 Range 对象，再通过 Range 对象的 Hidden 属性设置是否隐藏行或列。

```
Sub 隐藏行()
    r = ActiveCell.Row
    ws1.Rows(r).Hidden = True
End Sub
```

提示：如果设置 Hidden 属性为 False，则当前行将显示出来。

14.4.5　设置行高

要设置工作表中行的高度，需使用 Range 对象的 RowHeight 属性，该属性以磅为单位返回或设置指定区域中所有行的行高。如果指定区域中的各行的行高不等，则返回 Null。

如果要返回几行的 RowHeight 属性，可得到每一行的行高（如果所有的行等高），或得到 Null（如果它们不等高）。如果要返回几行的 Height 属性，将得到所有行高的总和。

例如，以下代码可设置选中区域的行高：

```
Sub 设置行高()
    Dim h As Long, r As Long, i As Integer, n As Integer
    Dim ws1 As Worksheet
    h = Application.InputBox(prompt:="请输入所选行的高度: ", _
        Title:="输入行高", Type:=1)
    Set ws1 = ActiveSheet
    n = Selection.Rows.Count
    r = ActiveCell.Row
    For i = 1 To n
        ws1.Rows(r + i - 1).RowHeight = h
    Next
    Set ws1 = Nothing
End Sub
```

执行以上代码，将弹出如图 14-6 所示的对话框，输入行高值后单击【确定】按钮，则选中单元格将被设置为指定的行高。如果在 Excel 工作表中选中了多个单元格区域，则所选每行的高度都将设置为新输入的高度。



图 14-6 【输入行高】对话框

技巧：使用 Range 对象的 AutoFit 方法，可更改区域中的列宽或行高以达到最佳匹配。

14.4.6　设置列宽

与设置行高类似，设置列宽需使用 Range 对象的 ColumnWidth 属性。

例如，以下代码可设置选中区域的列宽：

```
Sub 设置列宽()
    Dim w As Long, c As Long, i As Integer, n As Integer
    Dim ws1 As Worksheet
    w = Application.InputBox(prompt:="请输入所选列的宽度: ", _
        Title:="输入列宽", Type:=1)
    If w = 0 Then Exit Sub
```

```
    Set ws1 = ActiveSheet
    n = Selection.Columns.Count
    c = ActiveCell.Column
    For i = 1 To n
        ws1.Columns(c + i - 1).ColumnWidth = w
    Next
    Set ws1 = Nothing
End Sub
```

技巧：使用 Range 对象的 AutoFit 方法，可更改区域中的列宽或行高以达到最佳匹配。

14.5 管 理 批 注

在 Excel 中，用户可以通过插入批注来对单元格添加注释。可以编辑批注中的文字，也可以删除不再需要的批注。

14.5.1 插入批注

在 Excel 界面中，选择【审阅】选项卡，单击【批注】组中的【新建批注】按钮，可为当前单元格插入批注，还可通过该组中的相关按钮处理批注。

也可以通过 VBA 代码对批注进行管理。在 VBA 中，将每一个批注作为一个 Comment 对象来处理。每个工作表的 Comment 对象组成一个 Comments 集合，如果工作表中没有批注，则该集合就为空。

使用 AddComment 方法可在区域内添加批注。以下代码将在第一张工作表的单元格 E5 中添加批注。

```
Sub 添加批注()
    With Worksheets(1).Range("e5").AddComment
        .Visible = False
        .Text "批注日期：" & Date
    End With
End Sub
```

Text 是 Comment 对象的方法，所以不使用等号设置其批注值。

注意：如果单元格中已经添加了批注，使用 AddComment 方法再插入批注时将产生错误。

14.5.2 查看批注

工作表中的批注全部在 Comments 集合中，所以可以用 For Each 循环将其逐个显示出来。例如，以下代码显示第一个工作表中各批注的作者及批注的内容：

```
Sub 查看批注()
    Dim cm As Comment
    Dim i As Integer, j As Integer
    i = Worksheets(1).Comments.Count
    For Each cm In Worksheets(1).Comments
        j = j + 1
        MsgBox "第" & j & "条/共" & i & "条批注" & vbCrLf & _
            "作者:" & cm.Author & vbCrLf & "批注内容:" & cm.Text
    Next
End Sub
```

执行以上代码，将在 Excel 中显示提示对话框，并在该对话框中显示一条批注信息，单击【确定】按钮，将显示下一个批注。

提示：执行以上代码之前，应先在第一个工作表中插入几个批注。

14.5.3　隐藏/显示批注

工作表中的批注为 Comment 对象，所有 Comment 对象组成 Comments 对象集合。通过设置该对象的 Visible 属性可设置或获取批注对象的显示与隐藏。

例如，以下代码可切换选中单元格批注的显示或隐藏状态：

```
Sub 批注显示状态()
    Dim rng1 As Range, rng2 As Range
    On Error Resume Next
    Set rng1 = ActiveCell
    Set rng2 = ActiveSheet.Cells.SpecialCells(xlCellTypeComments)
    If rng2 Is Nothing Then Exit Sub
    If Not Application.Intersect(rng1, rng2) Is Nothing Then
        MsgBox "显示/隐藏当前单元格的批注。"
        rng1.Comment.Visible = Not (rng1.Comment.Visible)
    End If
End Sub
```

以上代码首先获取对活动单元格的引用，接着获取当前工作表中具有批注的单元格的引用。通过将活动单元格与有批注的单元格集合进行重叠运算，如果返回值为 True，则表示活动单元格具有批注，然后再对 Comment 对象的 Visible 属性取反，即可在显示与隐藏批注之间进行切换。

14.5.4　删除批注

使用 Comment 对象的 Delete 方法可删除指定批注。

例如，以下代码通过一个提示对话框逐条显示批注信息，提示用户是否删除该条批注：

```
Sub 删除批注()
    Dim cm As Comment, str1 As String
```

```
    Dim i As Integer, j As Integer
    i = Worksheets(1).Comments.Count
    For Each cm In Worksheets(1).Comments
        j = j + 1
        str1 = "第" & j & "条/共" & i & "条批注" & vbCrLf & _
            "作者:" & cm.Author & vbCrLf & "批注内容:" & cm.Text & vbNewLine
            & vbNewLine
        str1 = str1 & "单击【是】按钮将删除该批注！"
        If MsgBox(str1, vbQuestion + vbYesNo, "删除批注") = vbYes Then
            cm.Delete
        End If
    Next
End Sub
```

执行以上代码，将显示如图 14-7 所示的提示对话框，单击【是】按钮将删除显示的批注，单击【否】按钮将显示下一条批注信息。

技巧：使用 Range 对象的 ClearComments 方法，可清除指定区域的所有单元格批注。如果选中全部单元格，则可以清除工作表中的所有批注。若只希望删除部分批注，则需要先构造一个单元格区域对象，再调用 ClearComments 方法来删除。

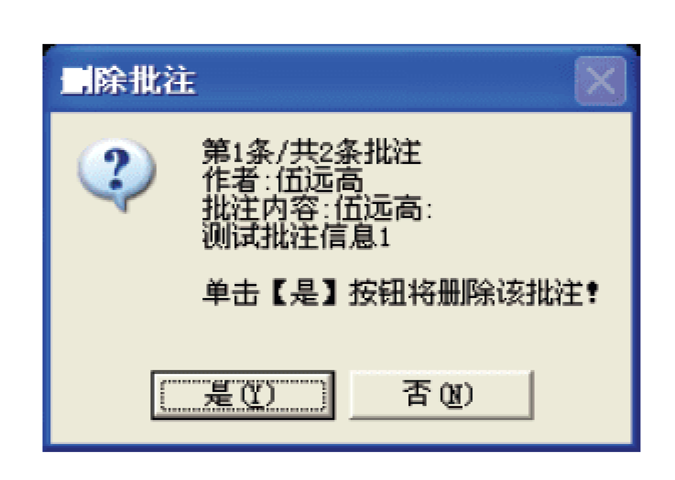


图 14-7 【删除批注】对话框

14.5.5 为输入数据的单元格添加批注

要为输入数据的单元格添加批注，就需要知道用户在哪个单元格中输入数据。这时，可使用 Worksheet 对象的 Change 事件来捕获工作表数据的更改。

Worksheet 对象的 Change 事件发生在用户更改工作表中的单元格或外部链接时，所引起的单元格的更改。其事件过程的语法格式如下：

```
Private Sub Worksheet_Change(ByVal Target As Range)

End Sub
```

其中，参数 Target 表示被修改的数据区域，是一个 Range 对象。通过该参数可获取数据被修改的单元格区域。

注意：当单元格在重新计算过程中更改时，将不会发生该事件。使用 Calculate 事件可以捕获工作表重新计算。

在工作表 Sheet1 的 Change 过程中编写以下 VBA 代码：

```
Private Sub Worksheet_Change(ByVal Target As Range)
    On Error Resume Next
    With Target
        .ClearComments
        If .Value <> "" Then
            .AddComment "" & Now
        Else
            Exit Sub
        End If
    End With
End Sub
```

当用户更改单元格的值后，如果目标单元格的值不为空，则为该单元格添加一个批注，批注的内容为当前的时间。

14.5.6　将原数据作批注

为了跟踪用户对工作表数据的修改，保存单元格原有的数据，可将单元格原有数据作为批注保存起来。

要将原有数据作为批注保存，可在 SelectionChange 事件过程中编写代码，在用户选择单元格时首先将该单元格的值保存到一个模块变量中。然后在 Change 事件过程将该模块变量的值作为批注的内容即可。

首先，在工作表 Sheet1 的声明部分定义一个模块变量 old，用来保存单元格的原值。

```
Dim old
```

其次，在工作表 Sheet1 的 SelectionChange 事件过程中编写以下代码，将单元格中的值保存到模块变量 old 中。

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    old = Target.Value   '保存单元格中的值
End Sub
```

最后，在工作表 Sheet1 的 Change 事件过程中编写以下代码，向单元格中添加批注。具体代码如下：

```
Private Sub Worksheet_Change(ByVal Target As Range)
    With Target
        If .Count > 1 Or .Value = "" Then Exit Sub   '多个单元格或值为空则退出
        If old <> "" Then
            .Comment.Visible = True
            .Comment.Text Text:=.Comment.Text & vbNewLine & _
                "于 " & VBA.Now & " 修改，原数据为： " & old '在原批注中添加内容
        Else
            .AddComment   '添加批注
```

```
            .Comment.Text Text:=.Comment.Text & "于 " & VBA.Now & " 修改"
        End If
    End With
End Sub
```

编写好以上代码后，在 Sheet1 工作表的单元格 A3 中输入数据，每次输入新的数据后，原数据将追加到批注中，如图 14-8 所示。

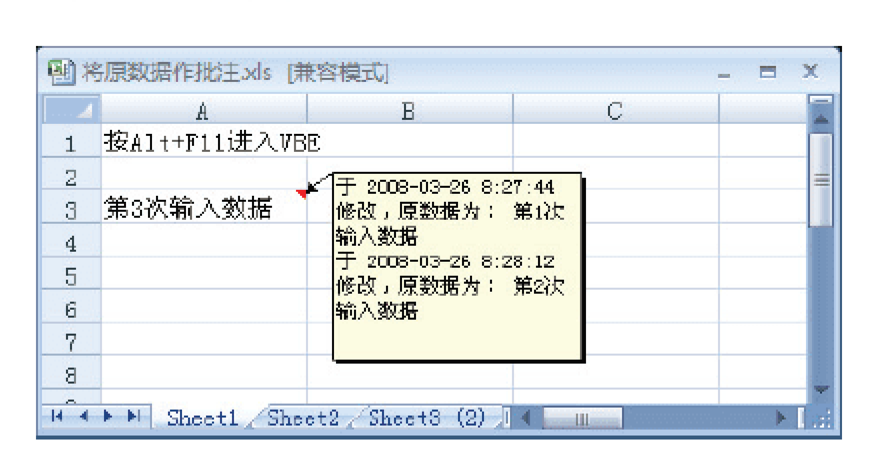


图 14-8　添加批注

14.6　操作单元格

用户可对单元格进行多种操作，例如，输入数据、删除数据、设置公式、复制数据、合并和拆分单元格等操作。

14.6.1　给单元格设置公式

可使用 Range 对象的以下两个属性来设置公式。

- Formula 属性：返回或设置一个 Variant 值，它代表 A1 样式表示法和宏语言中的对象的公式。
- FormulaR1C1 属性：返回或设置指定对象的公式，该公式使用宏语言 R1C1 格式符号表示。

例如，以下代码可为单元格 A1 设置相应的公式：

```
Sheet1.Range("A1").Formula = "=$A$4+$A$10"
Sheet1.Range("A1").Formula = "=SUM(B2:D2)"
Sheet1.Range("A1").FormulaR1C1 = "=SQRT(R1C1)"
```

14.6.2　复制公式

复制单元格公式的操作与复制单元格数据类似，首先将具有公式的单元格复制到剪贴板中，再使用 PasteSpecial 方法将 Range 从剪贴板粘贴到指定的区域中。PasteSpecial 方法的语法格式如下：

```
表达式.PasteSpecial(Paste, Operation, SkipBlanks, Transpose)
```

其中的各参数都可省略，其含义分别如下所述。

- Paste：设置要粘贴的内容，可使用 Excel 中的 xlPasteType 枚举数据，具体内容如表 14-1 所示。
- Operation：设置粘贴操作的方法，可使用 xlPasteSpecialOperation 枚举数据，具体内容如表 14-2 所示。
- SkipBlanks：该参数如果为 True，则不将剪贴板上区域中的空白单元格粘贴到目标区域中。默认值为 False。
- Transpose：该参数如果为 True，则在粘贴区域时转置行和列。默认值为 False。

表 14-1　xlPasteType枚举值

名　称	值	描　述
xlPasteAll	–4104	粘贴全部内容
xlPasteAllExceptBorders	7	粘贴除边框外的全部内容
xlPasteAllUsingSourceTheme	13	使用源主题粘贴全部内容
xlPasteColumnWidths	8	粘贴复制的列宽
xlPasteComments	–4144	粘贴批注
xlPasteFormats	–4122	粘贴复制的源格式
xlPasteFormulas	–4123	粘贴公式
xlPasteFormulasAndNumberFormats	11	粘贴公式和数字格式
xlPasteValidation	6	粘贴有效性
xlPasteValues	–4163	粘贴值
xlPasteValuesAndNumberFormats	12	粘贴值和数字格式

表 14-2　xlPasteSpecialOperation枚举值

名　称	值	描　述
xlPasteSpecialOperationAdd	2	复制的数据与目标单元格中的值相加
xlPasteSpecialOperationDivide	5	复制的数据除以目标单元格中的值
xlPasteSpecialOperationMultiply	4	复制的数据乘以目标单元格中的值
xlPasteSpecialOperationNone	–4142	粘贴操作中不执行任何计算
xlPasteSpecialOperationSubtract	3	复制的数据减去目标单元格中的值

例如，以下代码将单元格 E2 中的公式分别粘贴到下方的 3 个单元格中：

```
Sub 复制公式()
   Dim i As Integer
   With Sheet1
       .Range("E2").Copy
      For i = 3 To 5
          .Range("E" & i).PasteSpecial Paste:=xlPasteFormulas
      Next
   End With
End Sub
```

14.6.3 给单元格设置错误值

Excel 单元格的错误值共有 7 个。工作表中有时需要为单元格设置错误值，这时可使用 CVErr 函数来完成该任务。CVErr 函数的语法格式如下：

```
CVErr(errornumber)
```

参数 errornumber 可以是任何有效的错误号代码。

可以在过程中使用 CVErr 函数来创建用户自定义的错误。例如，如果创建一个函数，使其可以接受若干个参数，并且正常返回一个字符串，那么就可以让该函数来判断输入的参数，确认它们是否在可接受的范围内。如果不是，则此函数将不会返回所期望的字符串。在这种情况下，CVErr 可以返回一个错误号，并告知应该采取的行动。

注意： Error 的隐式转换是不允许的，例如，不能直接把 CVErr 的返回值赋值给一个非 Variant 的变量。然而，可以对 CVErr 的返回值进行显式转换（使用 CInt、CDbl 等），并赋值给适当的数据类型变量。

例如，以下代码将在单元格中分别设置错误值：

```
Sub 设置错误值()
    Dim arr1, i As Integer
    arr1 = Array(xlErrNull, xlErrDiv0, xlErrValue, xlErrRef, _
        xlErrName, xlErrNum, xlErrNA)
    For i = 1 To 7
        ActiveSheet.Cells(8, i).Value = CVErr(arr1(i - 1))
    Next i
End Sub
```

以上代码首先定义一个包含错误值常量的数组，再使用循环将各错误值分别赋值给各单元格。

14.6.4 判断错误类型

使用 IsError 函数可返回表达式是否为一个错误值。其语法格式如下：

```
IsError(expression)
```

参数 expression 可以是任何有效表达式。

IsError 函数被用来确定一个数值表达式是否表示一个错误。如果 expression 参数表示一个错误，则 IsError 返回 True；否则返回 False。

例如，以下代码将判断活动单元格的错误类型：

```
Sub 判断错误类型()
    Dim err_val
    If IsError(ActiveCell.Value) Then
```

```
        err_val = ActiveCell.Value
        Select Case err_val
            Case CVErr(xlErrDiv0)
                MsgBox "#DIV/0! error"
            Case CVErr(xlErrNA)
                MsgBox "#N/A error"
            Case CVErr(xlErrName)
                MsgBox "#NAME? error"
            Case CVErr(xlErrNull)
                MsgBox "#NULL! error"
            Case CVErr(xlErrNum)
                MsgBox "#NUM! error"
            Case CVErr(xlErrRef)
                MsgBox "#REF! error"
            Case CVErr(xlErrValue)
                MsgBox "#VALUE! error"
            Case Else
                MsgBox "不可识别错误! "
        End Select
    End If
End Sub
```

以上代码首先判断活动单元格，如果单元格中为错误值，则使用 CVErr 函数转换各错误值的常量，与单元格中的值进行判断，并显示出错误类型。

14.6.5　设置打印区域

Excel 工作表中可通过设置打印区域，使工作表中的部分数据进行打印输出。但大多数时候都需要将当前工作表所有使用区域设置为打印区域。

可使用 PageSetup 对象的 PrintArea 属性设置打印区域。

PrintArea 属性以字符串返回或设置要打印的区域，该字符串使用宏语言的 A1 样式的引用。将该属性设置为 False 或空字符串（""），可打印整个工作表。

提示： PageSetup 对象包含所有页面设置的属性（左边距、底部边距、纸张大小等）。

例如，使用以下代码可将当前工作表的使用区域设置为打印区域：

```
Sub 设置打印区域()
    Dim str1 As String
    str1 = ActiveSheet.UsedRange.Address
    ActiveSheet.PageSetup.PrintArea = str1
    str1 = ""
End Sub
```

以上代码首先取得当前使用区域地址，再使用 PrintArea 属性设置打印区域。

14.6.6 合并单元格

使用 Range 对象的 Merge 方法可创建合并单元格，该方法的语法格式如下：

```
表达式.Merge(Across)
```

其中的参数 Across 可省略，该参数如果为 True，则将指定区域中每一行的单元格合并为一个单独的合并单元格。默认值是 False（将区域中的多行单元格合并为一个单元格）。

提示：合并单元格的值在合并单元格后的左上角单元格中指定。

例如，使用以下代码可将单元格 A1 和 A2 合并为一个单元格：

```
Range("A1", "A2").Merge
```

而以下代码将单元格区域 A1:C3 共 9 个单元格合并为一个单元格：

```
Range("A1", "C3").Merge
```

14.6.7 拆分单元格

使用 Range 对象的 UnMerge 方法，可以将合并区域分解为独立的单元格。对非合并区域使用 UnMerge 方法将不会产生任何作用。也可以使用 MergeCells 属性判断指定区域是否包含合并单元格，如果包含合并单元格，则该属性返回值为 True。

例如，以下代码对选中的单元格区域进行拆分：

```
Sub 拆分单元格()
    Dim rng As Range
    If Not Selection.MergeCells Then
        MsgBox "选中区域不是合并区域！"
        Exit Sub
    End If

    Selection.UnMerge                           '拆分单元格
    For Each rng In Selection                   '逐个处理选中区域的单元格
        If rng = "" Then                        '若当前单元格为空
            rng = rng.Offset(-1, 0).Value       '取上一单元格的值
        End If
    Next
End Sub
```

以上代码首先判断用户选择的单元格是否为合并区域，如果是合并区域，则调用 UnMerge 方法拆分单元格，接着循环处理拆分的每一个单元格，将其赋值为上一个单元格的值。

14.6.8　限制单元格移动范围

使用 Worksheet 对象的 ScrollArea 属性，可设置允许用户滚动的区域。其参数为一个以 A1 样式的区域引用形式表示的区域。用户不能选定滚动区域之外的单元格。

如果将该属性设置为空字符串（""），则表示允许对整张工作表内所有单元格的选定（相当于取消对滚动区域的限制）。

也可通过 ScrollArea 属性获取当前滚动区域的设置值。

例如，使用以下代码可设置单元格移动范围为当前数据区域。

```
Sub 限制移动范围()
    Dim rng1 As Range, str1 As String
    Set rng1 = ActiveCell.CurrentRegion
    str1 = rng1.Address
    MsgBox "将单元格的移动范围限制在单元格当前区域 " & str1 & " 之内"
    ActiveSheet.ScrollArea = str1
End Sub
```

以上代码先获取活动单元格当前区域的地址，再通过 ScrollArea 属性设置滚动区域。执行以上代码后，用户将不能再将单元格区域滚动到其他单元格。

在需要的时候，可使用以下代码解除移动范围限制：

```
Sub 解除范围限制()
    MsgBox "解除移动范围限制"
    ActiveSheet.ScrollArea = ""
End Sub
```

以上代码设置 ScrollArea 属性为空字符串，就可以解除移动范围限制。

14.6.9　清除单元格

可以使用 Range 对象的多种方法清除单元格，各方法的功能分别如下所述。

- ClearContents 方法：清除单元格区域中的内容公式。
- ClearFormats 方法：清除单元格区域中的格式设置。
- ClearComments 方法：清除单元格区域中的所有单元格批注。
- Clear 方法：清除单元格区域的公式和格式设置。

例如，以下代码将清除工作表 Sheet1 上 A1:G37 单元格区域的公式，但保留其格式设置。

```
Sheet1.Range("A1:G37").ClearContents
```

以下代码清除工作表 Sheet1 上 A1:G37 单元格区域的所有格式设置。

```
Sheet1.Range("A1:G37").ClearFormats
```

以下代码清除工作表 Sheet1 上第一个嵌入式图表的格式设置。

```
Sheet1.ChartObjects(1).Chart.ChartArea.ClearFormats
```

以下代码清除 E5 单元格的所有批注。

```
Sheet1.Range("E5").ClearComments
```

14.6.10　删除单元格区域

在 Excel 环境中，在功能区【开始】选项卡的【单元格】组中，单击【删除单元格】命令按钮，将打开如图 14-9 所示的对话框，可根据需要选择替换删除单元格的方向。



图 14-9　删除单元格

在 VBA 中，使用 Range 对象的 Delete 方法也可完成删除选中单元格区域的功能，其语法格式如下：

```
表达式.Delete(Shift)
```

参数 Shift 可省略，用来设置如何调整单元格以替换删除的单元格。如果省略此参数，Excel 将根据区域的形状确定调整方式。该参数可设置为以下常量之一。

- xlShiftToLeft：单元格向左移动替换被删除的单元格。
- xlShiftUp：单元格向上移动替换被删除的单元格。

例如，以下代码将删除选中的单元格区域，单元格随之向左移动替换被删除的单元格。

```
Sub 删除单元格()
    Dim rng1 As Range
    Set rng1 = Selection
    rng1.Delete (xlShiftToLeft)
    Set rng1 = Nothing
End Sub
```

14.7　设置单元格格式

在 Excel 中进行操作时，很多时候都需要对单元格进行设置，以制作出美观大方的表格。在 VBA 中，也可使用各种命令进行单元格的格式设置。

14.7.1　设置自动套用格式

在 Excel 2003 及以前版本中，Range 对象提供了一个 AutoFormat 的方法，使用该方法可对选中区域自动套用格式。在 Excel 2007 版本中该方法被隐藏了，但仍然可以使用。该方法的语法格式如下：

```
表达式.AutoFormat(Format, Number, Font, Alignment, Border, Pattern, Width)
```

该方法各参数都可省略，各参数的含义分别如下所述。

- Format：指定的自动套用格式，可设置为 xlRangeAutoFormat 常量之一，默认常量为 xlRangeAutoFormatClassic1（古典 1 样式），该类常量非常多，使用时可查看帮助中的信息。
- Number：如果该值为 True，则在自动套用格式中包括数字格式。默认值为 True。
- Font：如果该值为 True，则在自动套用格式中包括字体格式。默认值为 True。
- Alignment：如果该值为 True，则在自动套用格式中包括对齐方式。默认值为 True。
- Border：如果该值为 True，则在自动套用格式中包括边框格式。默认值为 True。
- Pattern：如果该值为 True，则在自动套用格式中包括图案格式。默认值为 True。
- Width：如果该值为 True，则在自动套用格式中包括列宽和行高。默认值为 True。

例如，以下代码给工作表 Sheet1 的数据区域设置自动套用格式：

```
Sub 自动套用格式()
    Dim rng1 As Range
    Set rng1 = Sheet1.Range("A1").CurrentRegion
    rng1.autoformat
    Set rng1 = Nothing
End Sub
```

14.7.2　设置边框线

设置所选区域的边框可通过 Borders 集合来进行设置。通过 Range 对象的 Borders 属性可返回一个 Borders 集合，该集合代表样式或单元格区域（包括定义为条件格式一部分的区域）的边框。

Borders 集合由 4 个 Border 对象组成，分别代表 Range 对象的 4 个边框，即左边框、右边框、顶部边框和底部边框。

使用以下形式可返回单个 Border 对象：

```
表达式.Borders(index)
```

其中参数 index 用来指定边框，可分别用以下常量表示不同的边框。

- xlDiagonalDown：从区域中每个单元格的左上角至右下角的边框。
- xlDiagonalUp：从区域中每个单元格的左下角至右上角的边框。
- xlEdgeBottom：区域底部的边框。

- xlEdgeLeft：区域左边的边框。
- xlEdgeRight：区域右边的边框。
- xlEdgeTop：区域顶部的边框。
- xlInsideHorizontal：区域中所有单元格的水平边框（区域以外的边框除外）。
- xlInsideVertical：区域中所有单元格的垂直边框（区域以外的边框除外）。

例如以下代码设置单元格区域“A1:G1”的底部边框的颜色。

```
Sheet1.Range("A1:G1").Borders(xlEdgeBottom).Color = RGB(255, 0, 0)
```

以下代码设置当前区域的边框线为双线。

```
Sub 设置边框线()
    Dim rng1 As Range
    Set rng1 = Sheet1.Range("A1").CurrentRegion
    rng1.Borders.LineStyle = xlDouble
    Set rng1 = Nothing
End Sub
```

14.7.3　设置文本对齐格式

在 Excel 中可以通过【设置单元格格式】对话框设置单元格中文本的水平和垂直对齐方式。使用 VBA 设置单元格区域的对齐方式时，可使用 Range 对象的 HorizontalAlignment 属性和 VerticalAlignment 属性来进行设置。

1. HorizontalAlignment属性

使用该属性可返回或设置指定单元格区域的水平对齐方式。此属性的值可设为以下常量之一。

- xlCenter：表示居中；
- xlDistributed：表示分散对齐；
- xlJustify：表示两端对齐；
- xlLeft：表示左对齐；
- xlRight：表示右对齐。

2. VerticalAlignment属性

使用该属性可返回或设置指定单元格区域的垂直对齐方式。此属性的值可设为以下常量之一。

- xlVAlignBottom：表示靠下；
- xlVAlignCenter：表示居中对齐；
- xlVAlignDistributed：表示分散对齐；
- xlVAlignJustify：表示两端对齐；
- xlVAlignTop：表示靠上。

例如，使用以下代码可设置选择区域的单元格文本对齐方式为“水平居中”：

```
Selection.HorizontalAlignment = xlHAlignCenter
```

而以下代码设置为“垂直居中”：

```
Selection.VerticalAlignment = xlVAlignCenter
```

14.7.4　单元格文本缩排

在 VBA 中设置单元格的缩排值时，可使用 IndentLevel 属性来获取或设置缩进值，也可使用 InsertIndent 方法增加或减少缩进值。

1．IndentLevel属性

通过该属性可返回或设置单元格或单元格区域的缩进量。可以是 0～15 之间的整数。若将此属性设置为小于 0 或者大于 15 的数字，将导致程序发生错误。

2．InsertIndent方法

使用 Range 对象的 InsertIndent 方法，可向指定的区域添加缩进量。其语法格式如下：

```
表达式.InsertIndent(InsertAmount)
```

参数 InsertAmount 为设置的缩进量。若设置为负数，则可减少缩进量。

如果用本方法将缩进量设置为一个小于 0（零）或大于 15 的值，将导致程序出错。

例如，使用以下代码可为选择区域各单元格增加缩排值：

```
Sub 增加缩排值()
    On Error Resume Next
    Selection.InsertIndent 1
End Sub
```

其中，使用 On Error 语句可捕获因设置大于（大于 15）缩排值时产生的错误。

使用以下代码可减少单元格文本的缩排值：

```
Sub 减少缩排值()
    On Error Resume Next
    Dim rng1 As Range
    Set rng1 = Selection
    If rng1.IndentLevel > 0 Then
        rng1.InsertIndent -1
    End If
    Set rng1 = Nothing
End Sub
```

使用 InsertIndent 方法减少缩排值时，如果缩排值已经为 0，当再次执行 InsertIndent(–1)时，程序将出现错误。为了防止这种错误出现，可先使用 Range 对象的 IndentLevel 属性获取单元格区域的缩进值，若该值大于 0，则可以执行减少缩排操作。

14.7.5 设置文本方向

通过 Range 对象的 Orientation 属性，可获取或设置单元格区域文字的方向。其值可以从–90°～90°的一个整数值或以下常量之一。

- xlDownward：表示文字向下排列；
- xlHorizontal：表示文字水平排列；
- xlUpward：表示文字向上排列；
- xlVertical：表示文字在单元格中向下居中排列。

例如，使用以下代码，可设置文本方向为任意角度：

```
Sub 设置文本方向()
    Dim i As Integer
    i = Application.InputBox(prompt:="输入文字的角度(-90°～90°):", Type:=1)
    If i >= -90° And i <= 90° Then
        Selection.Orientation = i
    End If
End Sub
```

执行以上代码，将显示如图 14-10 所示的对话框，输入文字的角度后，选择区域的文本将按设置的角度显示。

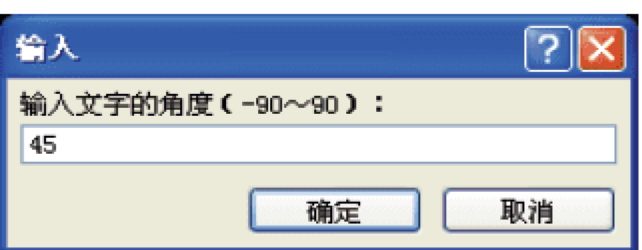


图 14-10 输入文本角度

14.7.6 设置自动换行格式

使用 Range 对象的 WrapText 属性，可获取或设置是否为单元格区域中的文本设置了自动换行。

如果指定区域内所有单元格中的文本都自动换行，此属性将返回 True；如果指定区域内所有单元格中的文本都不自动换行，则返回 False；如果指定区域内有些单元格中的文本自动换行，而另一些单元格中的文本不自动换行，则返回 Null。

例如，使用以下代码可设置选择的单元格自动换行：

```
Sub 自动换行()
    Selection.WrapText = True
End Sub
```

14.7.7 设置缩小字体填充

使用 Range 对象的 ShrinkToFit 属性，可获取或设置单元格区域的文本自动缩小，以适

应单元格列宽。

如果文本自动缩小以适应可用列宽，此属性将返回 True；如果没有将指定区域中所有单元格的这一属性设为相同的值，则返回 Null。

例如，使用以下代码可设置选中的单元格缩小字体，以便将内容完全显示在单元格中：

```
Sub 缩小字体填充()
    Selection.ShrinkToFit = True
End Sub
```

14.7.8　设置日期格式

使用 Range 对象的 NumberFormatLocal 属性，可以字符串的形式返回或设置一个代表单元格对象的格式代码。

日期格式有很多种形式，读者可通过录制宏，记录设置单元格日期格式，查看各种日期格式和格式代码。

例如，使用以下代码可将单元格日期格式设置为“年/月/日”的格式：

```
Sub 设置日期格式()
    Dim rng As Range, rng1 As Range
    Set rng1 = ActiveSheet.UsedRange
    For Each rng In rng1
        If IsDate(rng.Value) Then
            rng.NumberFormatLocal = "yyyy""年""m""月""d""日"";@"
        End If
    Next
End Sub
```

以上代码检查当前工作表的所有单元格，如果单元格的值为日期值，则将其转换为设定的格式。使用 IsDate 函数可判断一个表达式是否可以转换成日期。其语法格式如下：

```
IsDate(expression)
```

参数 expression 是一个日期表达式或字符串表达式，这里的字符串表达式是可以作为日期或时间来认定的。

如果表达式是一个日期，或可以作为有效的日期识别，则 IsDate 返回 True；否则返回 False。

14.7.9　生成大写金额

金额大写作为财务管理中的一种常用数据表示方式，在很多管理系统中都要用到。中文版 Excel 也支持阿拉伯数字转换为中文大写形式。通过录制宏可以看出，设置数字为中文大写样式的格式代码如下：

```
[DBNum2][$-804]G/通用格式
```

可是，这种中文大写样式不符合金额的要求，其转换后的效果如下：

```
壹仟零伍拾壹.壹
```

这种形式还不能作为财务金额大写来使用，要将以上内容转换为符合要求的金额大写形式还需要进行一些处理。

以下代码就可以完成这种转换。

```
Sub 大写金额()
    Dim t As Currency, str1 As String
    Dim i As Integer, strJ As String, strF As String
    Dim rng1 As Range
    With ActiveSheet
        Set rng1 = Range("IV1").End(xlToRight)   '获取最右侧列
        t = ActiveCell.Value
        With rng1
            .Value = t
            .NumberFormatLocal = "[DBNum2][$-804]G/通用格式"
            .Columns.AutoFit
            str1 = .Text
            .Clear
        End With
        i = InStr(str1, ".")
        If i > 0 Then
            strJ = Mid(str1, i + 1, 1)              '获取角部分字符
            strF = Mid(str1, i + 2, 1)              '获取分部分字符
            If strF = "" Then
                str1 = Left(str1, i - 1) & "元" & strJ & "角整"
            Else
                str1 = Left(str1, i - 1) & "元" & strJ & "角" & strF & "分"
            End If
        Else
            str1 = str1 & "元整"
        End If
        ActiveCell = "人民币" & str1
    End With
End Sub
```

以上代码使用第一行的最后一列作为临时单元格，将当前单元格的值填写到临时单元格中，再将临时单元格设置为中文大写样式，通过 Text 属性获取该单元格的显示值。最后通过字符串处理函数对中文大写字符进行处理。

使用 Range 对象的 Value 属性可取得单元格的值，以上代码需要取得转换后显示的中文大写字符串，但使用 Value 属性只能取得单元格中的数值，而不能获取显示的大写字符串。这时可使用 Range 对象的 Text 属性，该属性返回或设置指定对象中的文本。

接着将获取的文本使用字符串处理函数 Left 和 Mid，分别取出元、角、分各部分内容，再判断角和分部分是否为空，从而生成不同的大写金额字符串。

14.7.10　设置单元格图案

使用 Interior 对象的以下属性可设置单元格区域的内部图案和图案颜色。

- Pattern 属性：可获取或设置一个代表内部图案的值。
- PatternColor 属性：将以 RGB 值返回或设置内部图案的颜色。

例如，以下代码循环显示不同的内部图案，再使用随机函数 Rnd 生成图案的颜色并显示。

```
Sub 设置单元格图案()
    Dim i As Integer
    Dim r As Integer, g As Integer, b As Integer
    Randomize
    On Error Resume Next
    For i = 1 To 18
        With Selection.Interior
            .Pattern = i
            r = Int(Rnd * 255)
            g = Int(Rnd * 255)
            b = Int(Rnd * 255)
            .PatternColor = RGB(r, g, b)
        End With
        MsgBox "下个图案样式"
    Next i
End Sub
```

提示：设置 PatternColorIndex 属性为常量 xlColorIndexNone，表示为无色。

14.8　设置条件格式

在 Excel 中，FormatConditions 对象代表一个区域内所有条件格式的集合。FormatConditions 集合可以包含多个条件格式。每个格式由一个 FormatCondition 对象代表。

使用 FormatConditions 对象的 Add 方法，可向集合中添加新的条件格式。其语法格式如下：

```
表达式.Add(Type, Operator, Formula1, Formula2)
```

各参数的含义如下：

- Type：指定条件格式是基于单元格值还是基于表达式，可使用的常量如表 14-3 所示。
- Operator：条件格式运算符。如果 Type 为 xlExpression，则忽略 Operator 参数。可使用的常量如表 14-4 所示。
- Formula1：与条件格式相关联的值或表达式。可以是常量值、字符串值、单元格

引用或公式。

- ❑ Formula2：当参数 Operator 为 xlBetween 或 xlNotBetween 时，它是与条件格式第二部分相关联的值或表达式（否则忽略该参数）。可为常量值、字符串值、单元格引用或公式。

表 14-3 Type可用常量

xlAboveAverageCondition	高于平均值条件	xlIconSet	图标集
xlBlanksCondition	空值条件	xlNoBlanksCondition	无空值条件
xlCellValue	单元格值	xlNoErrorsCondition	无错误条件
xlColorScale	色阶	xlTextString	文本字符串
xlCompareColumns	比较列	xlTimePeriod	时间段
xlDatabar	数据条	xlTop10	前10个值
xlErrorsCondition	错误条件	xlUniqueValues	唯一值
xlExpression	表达式		

表 14-4 Operator可用常量

xlBetween	介于	xlNotBetween	不介于
xlEqual	等于	xlNotEqual	不等于
xlGreater	大于	xlGreaterEqual	大于等于
xlLess	小于	xlLessEqual	小于等于

向单元格区域添加条件格式的代码如下：

```
With 表达式.FormatConditions.Add(参数)
    设置格式代码
End With
```

使用 FormatConditions 集合对象的 Modify 方法可修改现有的条件格式，使用 Delete 方法可在添加新条件格式前删除现有的格式。

注意：对单个区域定义的条件格式不能超过 3 个。

例如，使用以下代码可为工作表 Sheet1 的区域“F32:F13”设置条件格式，当该区域中单元格的值大于等于 2000 时，显示为红色；当单元格的值小于 1000 时，显示为绿色。

```
Sub 设置条件格式()
    Dim rng1 As Range
    Set rng1 = Sheet1.Range("C2:E6")
    '添加条件格式，设置单元格值大于等于 2000 的格式
    With rng1.FormatConditions.Add(Type:=xlCellValue, _
       Operator:=xlGreaterEqual, Formula1:=2000)
        With .Borders
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = 6
        End With
        With .Font
```

```
            .Bold = True
            .ColorIndex = 3
        End With
    End With
    '添加条件格式，设置单元格值小于 1000 的格式
    With rng1.FormatConditions.Add(Type:=xlCellValue, _
       Operator:=xlLess, Formula1:=1000)
        With .Font
            .Bold = True
            .ColorIndex = 10
        End With
    End With
End Sub
```

对于当前工作表中的条件格式，可使用 FormatConditions 集合对象的 Delete 方法将其删除，代码如下：

```
Sub 清除条件格式()
    Cells.FormatConditions.Delete
End Sub
```

以上代码使用 Cells 返回当前工作表的全部单元格，通过上面的代码可清除当前工作表的所有条件格式。

第 15 章　其他常用 Excel 对象

除了前面几章介绍的 Excel 对象外，在开发 Excel 应用程序时还经常使用到 Name 对象、Window 对象、Chart 对象等。本章介绍用 VBA 控制这几个对象的方法。

15.1　使用 Name 对象

Name 对象代表单元格区域的定义名。名称可以是内置名称（例如 Database、Print_Area 和 Auto_Open）或自定义名称。

15.1.1　添加名称

在 Excel 中，可使用行号列标来引用单元格（如 A1），也可对单元格区域进行命名，然后在公式或 VBA 代码中使用名称来引用相关单元格。单元格区域的名称定义保存在 Name 对象中。

Name 对象是 Application、Workbook 和 Worksheet 对象的 Names 集合的成员。使用 Names(index)（其中 index 是名称索引号或定义名称）可返回一个 Name 对象。

注意：这里的 Name 对象，不是 Workbook 对象的 Name 属性。

可用 Add 方法创建名称并将其添加到集合中。下面的语句创建一个新名称，指向工作表 Sheet1 上单元格区域 A1:C20。

```
Names.Add Name:="test", RefersTo:="=sheet1!$a$1:$c$20"
```

RefersTo 参数必须以 A1 样式表示法指定，包括必要时使用的美元符号($)。例如，如果在工作表 Sheet1 上选定了单元格 A10，然后又将 RefersTo 参数指定为“=Sheet1!A1:B1”而定义了一个名称，那么该名称实际上指向单元格区域 A10:B10（因为指定的是相对引用）。若要指定绝对引用，应当用“=Sheet1!$A$1:$B$1”。

例如，在一个“库存管理系统”中，有一个名称为“商品信息”的工作表，如图 15-1 所示。在该工作表中的数据不断变化，如果需要对商品信息区域进行命名，供其他表格使用，可使用以下代码进行定义：

```
Sub 定义名称()
    Dim intRow As Integer
```

```
    intRow = Sheets("商品信息").[B65536].End(xlUp).Row '获取商品信息的数据行数
    ActiveWorkbook.Names.Add Name:="SP", _
        RefersToR1C1:="=商品信息!R3C1:R" & intRow & "C6"
                                                            '在"商品信息"表中定
                                                            义名称 SP
End Sub
```

对于已定义 Name 对象的名称，也可通过 VBA 代码进行修改。例如，在工作簿中已经定义了一个 Name，名称为 SP。

在【公式】选项卡的【定义的名称】组中，单击【名称管理器】按钮，打开【名称管理器】对话框，查看结果如图 15-2 所示。

商品信息					
商品编码	商品名称	规格型号	颜色	单位	单价
S001	三星手机	E508	黑	台	3500.00
N001	诺基亚手机	3200	灰	台	2800.00
N002	诺基亚手机	7260	红	台	3200.00
S002	三星手机	808	灰	台	1000.00
M001	摩托罗拉手机	1200	灰	台	3880.00
M002	摩托罗拉手机	V3	黑	台	1560.00
M003	摩托罗拉手机	V31	黑	台	2200.00

图 15-1　工作表

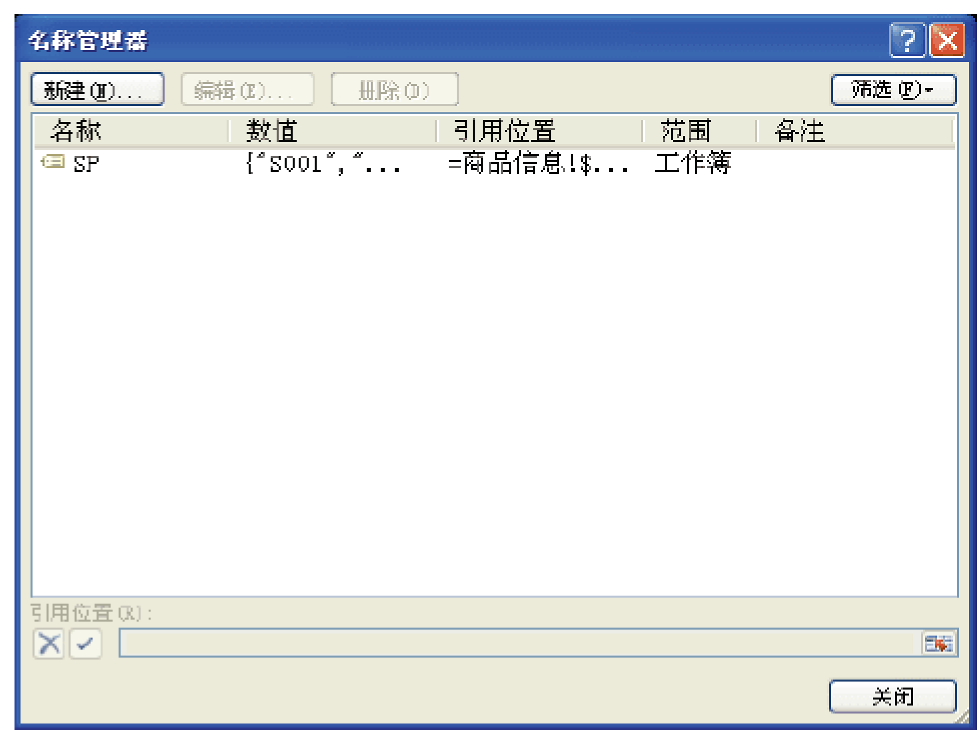


图 15-2　【名称管理器】对话框

15.1.2　修改名称

使用 Name 对象的 Name 属性可修改已有 Name 对象的名称。

🔔注意：这里两个 Name 关键字的意义是不同的，前一个 Name 表示对象，后一个 Name 表示对象的属性。

编写以下代码，可将名为 SP 的名称改为 SP1：

```
Sub 改变名称()
    Dim MyName As Name
    For Each MyName In Names
        If MyName.Name = "SP" Then
            MyName.Name = "SP1"
        End If
    Next
End Sub
```

执行以上过程后，在【公式】选项卡的【定义的名称】组中单击【名称管理器】按钮，打开【名称管理器】对话框，可看到改名后的结果如图 15-3 所示。

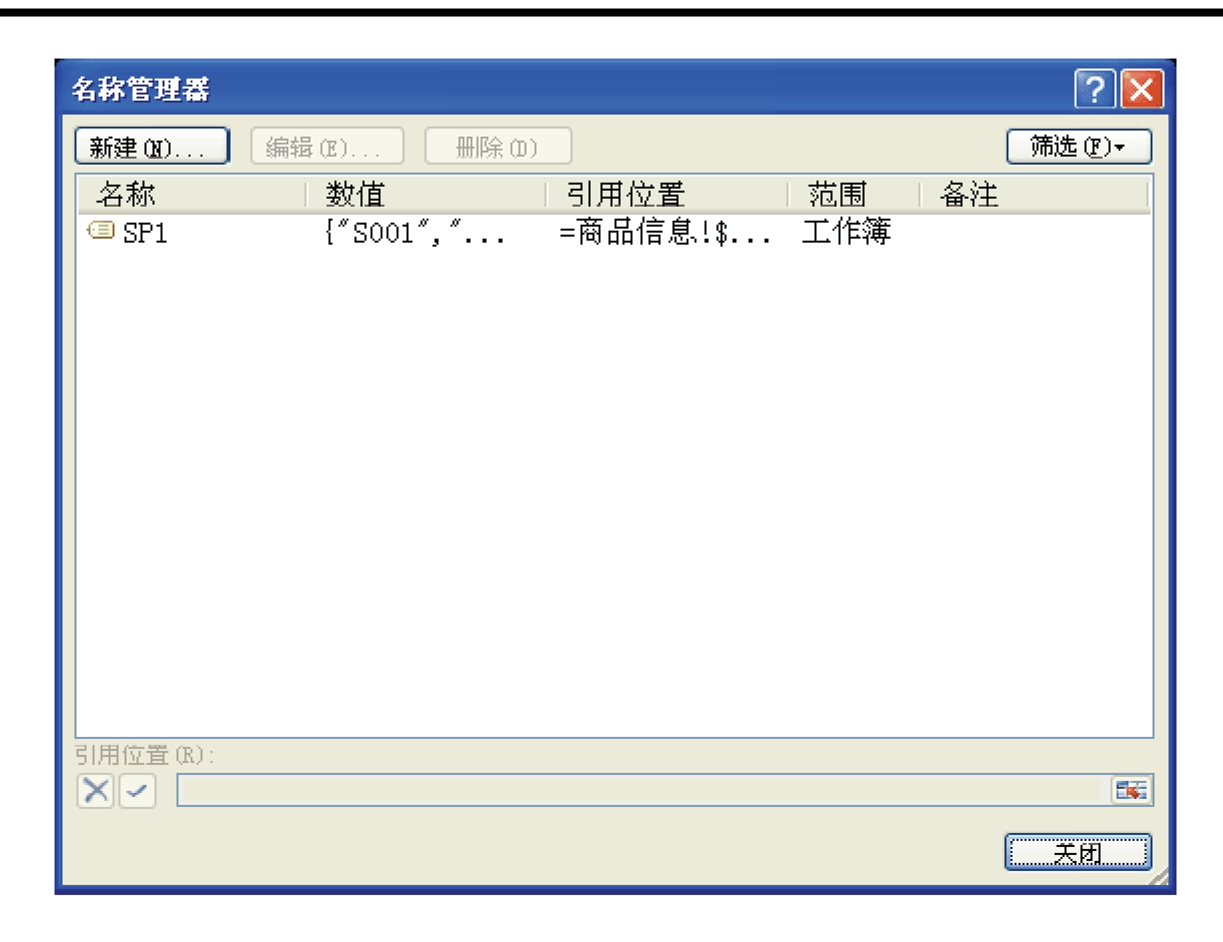


图 15-3　修改名称

15.1.3　显示名称的定义

在定义 Name 对象时，使用 RefersTo 参数表示 Name 对象引用的公式。在程序中，也可使用 RefersTo 属性，用宏语言以 A1 样式表示法返回或设置名称所引用的公式（以等号开头）。

例如，以下代码创建活动工作簿中所有名称的列表，并用宏语言以 A1 样式表示法表示这些名称所引用的公式。该列表将出现在一个新建的工作表中。

```
Sub 名称列表()
    Set NewSheet = Worksheets.Add                   '新建一个工作表
    i = 1
    For Each nm In ActiveWorkbook.Names             '循环处理工作簿中的 Name 对象
        NewSheet.Cells(i, 1).Value = nm.Name        'Name 对象的名称
        NewSheet.Cells(i, 2).Value = "'" & nm.RefersTo
                                                    'Name 对象的公式
        i = i + 1
    Next
    NewSheet.Columns("A:B").AutoFit
End Sub
```

提示： 除了使用 RefersTo 返回或设置指定名称所引用的公式外，还可使用 RefersToR1C1 属性，以 R1C1 样式返回或设置指定名称所引用的公式。

15.1.4　获取 Name 对象的引用

使用 RefersTo 或 RefersToR1C1 属性可返回 Name 对象所引用的公式，这些属性返回的值为字符串类型。若需要在 VBA 代码中引用 Name 对象中公式定义的单元格，可使用 RefersToRange 属性，该属性返回由 Name 对象引用的 Range 对象。

注意：如果 Name 对象并不引用区域（例如，该对象引用的是一个常量或公式），则该属性无效。

例如，使用以下代码使用 RefersToRange 属性获取工作簿中各名称的引用，再显示出该名称占用的单元格行、列、单元格数量。

```
Sub 查看名称区域信息()
    Dim nm As Name, str1 As String, rng1 As Range
    For Each nm In ActiveWorkbook.Names          '循环处理工作簿中的 Name 对象
        str1 = "Name 对象名称: " & nm.Name & vbNewLine
        str1 = str1 & "公式: " & nm.RefersTo & vbNewLine
        Set rng1 = nm.RefersToRange
        str1 = str1 & "该对象所占单元格数量: " & rng1.Cells.Count & vbNewLine
        str1 = str1 & "行数: " & rng1.Rows.Count & vbNewLine
        str1 = str1 & "列数: " & rng1.Columns.Count
        MsgBox str1, vbOKOnly, "查看名称区域信息"
    Next
End Sub
```

执行以上代码，将显示如图 15-4 所示的对话框，显示当前工作簿中某一个 Name 对象的相应信息。

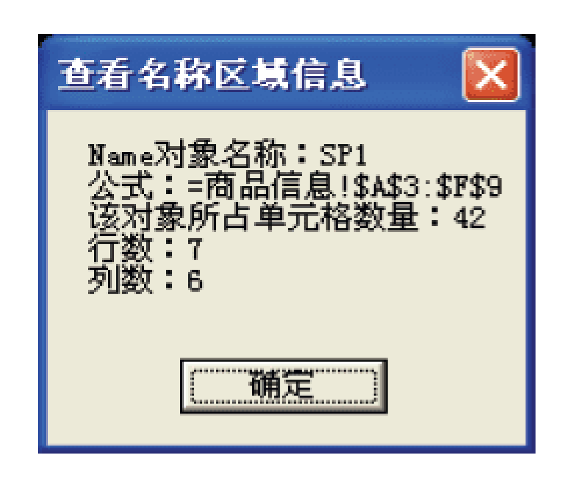


图 15-4 【查看名称区域信息】对话框

15.2　使用 Window 对象

Window 对象代表一个窗口，可以对窗口特性进行设置和操作。许多工作表特征（例如滚动条和标尺）实际上是窗口的属性。

Window 对象是 Windows 集合的成员。在 Excel 中，Application 对象和 Workbook 对象都有 Windows 集合，其中 Application 对象的 Windows 集合包含应用程序中的所有窗口，而 Workbook 对象的 Windows 集合只包含指定工作簿中的窗口。

15.2.1　创建窗口

在 Excel 2007 操作环境中，在【视图】选项卡的【窗口】组中，单击【新建窗口】按钮，可新建一个 Excel 窗口，该窗口的标题栏名称将显示为“Book1.xlsm:2”，在新建的窗

口中将显示活动窗口的副本，如图 15-5 所示。

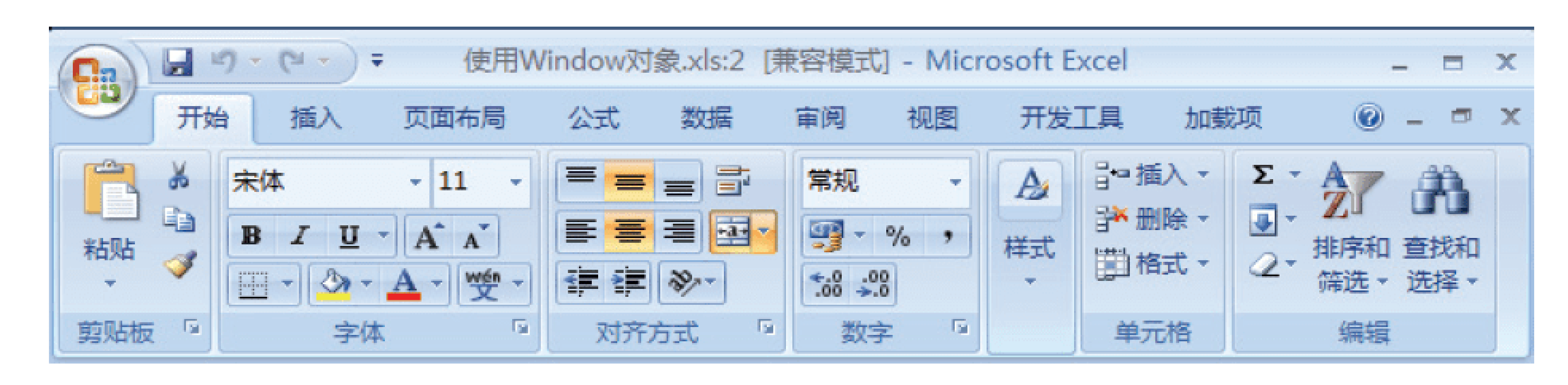


图 15-5　第一个窗口副本

在 VBA 代码中，使用 Window 对象的 NewWindow 方法，可新建一个窗口或者创建指定窗口的副本。

注意：窗口号和窗口索引（Index 属性）是两个不同的概念，例如，名称为"Book1.xlsm:2"的窗口，其窗口号为 2，而窗口索引为该窗口在 Windows 集合中的位置，可以是窗口名称或编号。

例如，以下代码将创建一个新窗口：

```
Sub 创建窗口()
    ActiveWindow.NewWindow
    MsgBox "新建窗口的窗口号是: " & ActiveWindow.WindowNumber
End Sub
```

Window 对象的 WindowNumber 属性返回窗口号。例如，名称为"Book1.xls:2"的窗口，其窗口号为 2。大多数窗口的窗口号为 1。

15.2.2　调整窗口大小

通过 Window 对象的 EnableResize 属性可控制是否能够调整窗口大小，如果其值为 True，则能够调整窗口大小；如果其值为 False，则不允许调整窗口大小。

控制窗口大小的属性主要有：控制窗口左上角位置的两个属性（Top 和 Left），控制窗口的宽度和高度的两个属性（Width 和 Height），各属性的含义如下所述。

- Top 属性：代表从窗口上边缘到使用区域（在菜单、任何停放在顶端的工具栏和编辑栏下方）上边缘的距离（以磅为单位）。
- Left 属性：代表从客户区左边缘到窗口左边缘的距离（以磅为单位）。
- Width 属性：代表窗口的宽度（以磅为单位）。
- Height 属性：代表窗口的高度（以磅为单位）。

注意：无法对最大化窗口设置这些属性。

例如，使用以下代码将动态改变窗口大小：

```
Sub 动态改变窗口大小()
```

```
    With ActiveWindow
        .WindowState = xlNormal
        .EnableResize = True
        .Top = 1
        .Left = 1
        .Height = 40
        .Width = 40
        For i = 40 To Application.UsableWidth
            .Width = i
        Next
        For i = 40 To Application.UsableHeight
            .Height = i
        Next
    End With
err1:
End Sub
```

执行以上代码，当前窗口将首先逐渐变宽，然后再逐步变高，达到动态改变窗口大小的效果。

注意：以上过程需要在 Excel 界面中执行，才能查看到动态的效果。

15.2.3　获取窗口状态

窗口的状态包括两个方面，首先是 Excel 应用程序窗口的状态，另一个是工作簿窗口的状态。窗口状态有 3 种形式，通过 Window 对象的 WindowState 属性可返回或设置窗口的状态。可用以下常量表示窗口的状态。

- xlMaximized：最大化；
- xlMinimized：最小化；
- xlNormal：正常。

使用以下代码可获取窗口状态：

```
Sub 获取窗口状态()
    Dim str1 As String
    str1 = "Excel 应用程序窗口的状态："
    Select Case Application.WindowState
        Case xlMaximized
            str1 = str1 & "最大化。"
        Case xlMinimized
            str1 = str1 & "最小化。"
        Case xlNormal
            str1 = str1 & "正常。"
    End Select
```

```
    MsgBox str1

    str1 = "当前活动工作簿窗口的状态："
    Select Case ActiveWindow.WindowState
        Case xlMaximized
            str1 = str1 & "最大化。"
        Case xlMinimized
            str1 = str1 & "最小化。"
        Case xlNormal
            str1 = str1 & "正常。"
    End Select
    MsgBox str1
End Sub
```

以上代码首先获取 Excel 应用程序窗口的状态并显示出来，接着获取工作簿窗口的状态并显示出来。

15.2.4 拆分窗格

通过 Window 对象的 Split 属性，可查询窗口是否被拆分。如果指定窗口被拆分，则该属性值为 True。将该属性值设置为 False，可取消指定窗口的拆分状态。

窗口可以进行水平和垂直两个方向上的拆分，可通过 Window 对象的以下两个属性进行控制。

- SplitRow 属性：返回或设置将指定窗口拆分成窗格处的行号（拆分线以上的行数）。
- SplitColumn 属性：返回或设置将指定窗口拆分成窗格处的列号（拆分线左侧的列数）。

使用以下代码可拆分窗格：

```
Sub 拆分窗格()
    Dim r As Long, c As Long
    r = ActiveCell.Row
    c = ActiveCell.Column
    With ActiveWindow
        If .Split Then
            .Split = False
        Else
            .SplitRow = r - 1
            .SplitColumn = c - 1
       End If
    End With
End Sub
```

以上代码中，因为 SplitRow 和 Splitcolumn 属性是从指定行数的上方和列数的左侧进

行拆分，所以需将活动单元格的行数和列数减 1。

对拆分的窗格还可通过设置 Window 对象的 FreezePanes 属性为 True 来冻结窗格，如果设置为 False，则可以取消冻结窗口。

例如，以下代码可在冻结和取消冻结之间转换：

```
ActiveWindow.FreezePanes = Not ActiveWindow.FreezePanes
```

15.2.5　设置窗口显示比例

在 Excel 2007 的状态栏中，双击右侧的 100% 图标，将打开如图 15-6 所示的【显示比例】对话框。通过该对话框，用户可以调整表格的显示比例。其比例以百分数表示（100 表示正常大小，200 表示双倍大小，依次类推）。

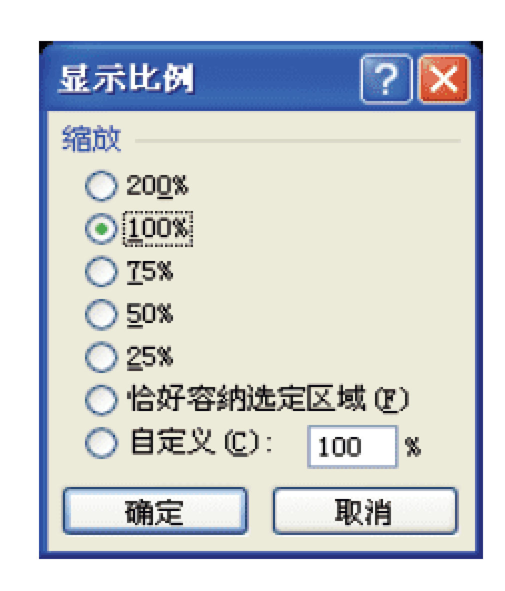


图 15-6　【显示比例】对话框

在 VBA 代码中，通过 Window 对象的 Zoom 属性来获取或设置窗口的显示比例。将此属性设为 True，可将窗口大小设置成与当前选定区域相适应的大小。

注意：本功能仅对窗口中当前的活动工作表起作用。如果要对其他工作表使用此属性，必须先激活工作表。

用 VBA 代码设置窗口显示比例的代码如下：

```
Sub 窗口显示比例()
    Dim s As Integer

    s = Application.InputBox(prompt:="请输入窗口的显示比例：" & _
        vbCrLf & "（100 表示正常大小，200 表示双倍大小，依次类推）。", _
        Title:="显示比例", Default:=100, Type:=1)
    If s = 0 Then Exit Sub
    ActiveWindow.Zoom = s
End Sub
```

运行以上代码，首先弹出如图 15-7 所示的对话框，要求用户输入显示比例，如果在该对话框中单击【取消】按钮，则返回值为 0，将退出子过程的执行。

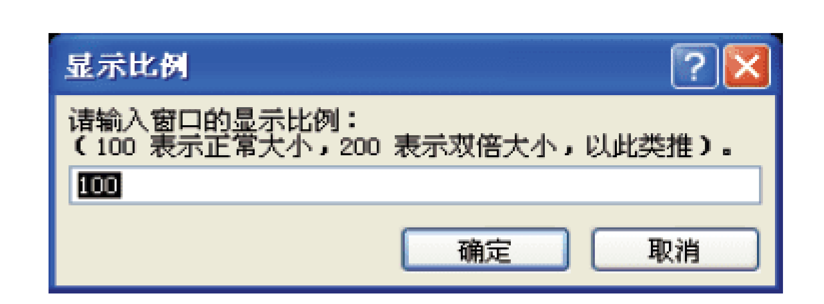


图 15-7　输入显示比例

15.2.6　设置工作簿显示选项

通过 Window 对象的以下 3 个属性值可获取或设置工作簿的显示选项。

- ❑ DisplayHorizontalScrollBar：水平滚动条；
- ❑ DisplayVerticalScrollBar：垂直滚动条；
- ❑ DisplayWorkbookTabs：工作表标签。

以上 3 个属性值如果为 True，则表示显示相关的元素，如果为 False 则表示隐藏相关的元素。

例如，以下代码分别对这 3 个属性值取反，即可在显示或隐藏之间进行切换。

```
    With ActiveWindow
        .DisplayHorizontalScrollBar = Not .DisplayHorizontalScrollBar
        .DisplayVerticalScrollBar = Not .DisplayVerticalScrollBar
        .DisplayWorkbookTabs = Not .DisplayWorkbookTabs
    End With
End Sub
```

通过 Window 对象的以下 3 个属性值可获取或设置工作表的显示选项。

- ❑ DisplayHeadings：显示行号列标；
- ❑ DisplayFormulas：显示公式；
- ❑ DisplayZeros：显示 0 值。

以上 3 个属性值如果为 True，则表示显示相关的元素，如果为 False 则表示隐藏相关的元素。

15.2.7　设置工作表网格线

使用 Window 对象的 DisplayGridlines 属性可获取或设置工作表是否显示网格线，如果该属性值为 True，则显示网格线。

提示： 该属性仅影响显示的网格线，不能控制网格线的打印。因此该属性仅适用于工作表和宏工作表。

网格线颜色可使用 Window 对象的以下两个属性进行设置。

- ❑ GridlineColor 属性：以 RGB 值返回或设置网格线颜色。
- ❑ GridlineColorIndex 属性：返回或设置网格线颜色，其值为当前调色板中的索引。

使用以下代码可设置网格线颜色：

```
Sub 设置网格线颜色()
    Dim i As Integer, r As Integer, g As Integer, b As Integer
    i = Application.InputBox(prompt:="请选择网格线的颜色: " & _
        vbCrLf & "（1.红色　2.绿色　3.蓝色）", _
        Title:="网格线颜色", Default:=1, Type:=1)
    Select Case i
        Case 1: r = 1
        Case 2: g = 1
        Case 3: b = 1
    End Select
    ActiveWindow.GridlineColor = RGB(r * 255, g * 255, b * 255)
End Sub
```

执行以上代码，将弹出如图 15-8 所示的对话框，输入 1～3 中的一个数，即可将网格线显示为红、绿、蓝色。

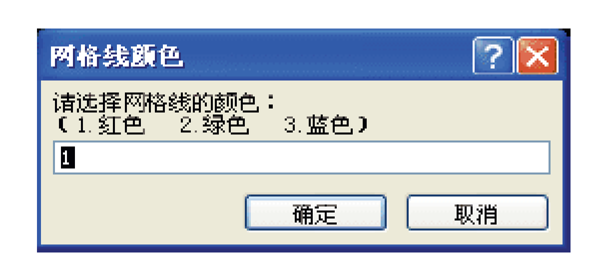


图 15-8　设置网格线颜色

15.3　使用 Chart 对象

在 Excel 中对数据进行分析时，使用图表可直观地查看分析结果。Excel 提供上百种图表类型，通过 VBA 代码可以控制图表的各个方面。

在 Excel 中可以快速简便地创建图表，在程序中，通过 VBA 代码也可方便地创建图表。在 Excel 中创建的图表，可以嵌入到工作表中数据的旁边，也可插入到一个新的图表工作表中，分别称为嵌入式图表和图表工作表。

15.3.1　创建图表工作表

用 VBA 创建图表工作表，一般按以下步骤进行：

（1）创建一个空的图表工作表。

Charts 集合包含工作簿中所有图表工作表的集合。每个图表工作表都由一个 Chart 对象来表示，但不包括嵌入在工作表或对话框编辑表上的图表。

通过 Charts 集合的 Add 方法可向集合中添加新的图表工作表（新建图表工作表），Add 方法的语法格式如下：

```
表达式.Add(Before, After, Count, Type)
```

该方法的参数都可省略，各参数的含义如下所述。

- ❑ Before：指定工作表的对象，新建的工作表将置于此工作表之前。
- ❑ After：指定工作表的对象，新建的工作表将置于此工作表之后。
- ❑ Count：要添加的工作表数。默认值为 1。
- ❑ Type：指定要添加的图表类型，可创建的图表类型很多，具体可参考 Excel VBA 的帮助信息。

提示：如果 Before 和 After 两者都被省略，新建的图表工作表将插入到活动工作表之前。

（2）设置数据源区域。

通过 Chart 对象的 SetSourceData 方法，可为指定图表设置源数据区域。其语法格式如下：

```
表达式.SetSourceData(Source, PlotBy)
```

该方法的两个参数含义如下所述。

- ❑ Source：为一个 Range 对象，用来指定图表的源数据区域。
- ❑ PlotBy：指定数据绘制方式。可使用常量 xlColumns（数据系列在行中）和 xlRows（数据系列在列中）之一。

（3）指定图表类型。

通过 Chart 对象的 ChartType 属性可获取或设置图表类型。

（4）设置图标标题。

通过 Chart 对象的 ChartTitle 属性，可返回一个 ChartTitle 对象，该对象表示指定图表的标题。通过该对象的属性可控制图表的标题，例如，设置标题文本、设置标题的格式等。

注意：只有图表的 HasTitle 属性为 True 时，ChartTitle 对象才存在，从而才能使用该对象。

以下代码根据工作表“成绩表”中的数据，生成簇状柱形图。

```
Sub 创建图表()
   Dim cht As Chart

   Set cht = Charts.Add     '创建图表对象
   With cht
      .SetSourceData Source:=Sheets("成绩表").Range("B2:E7"), PlotBy:=
      xlRows
                            '指定数据源
      .ChartType = xlColumnClustered

      .HasTitle = True      '添加标题
      .ChartTitle.Text = "成绩分析图"
   End With
End Sub
```

执行以上代码，将使用如图 15-9 左图所示的工作表“成绩表”中的数据，生成一个图表工作表，如图 15-9 右图所示。



	A	B	C	D	E	F
1			成绩表			
2	姓名	语文	数学	综合	总分	
3	张金山	65	97	90	252	
4	雷伟	83	95	83	261	
5	李军	67	68	94	229	
6	任小霞	91	93	85	269	
7	白松	84	90	90	264	
8	林小平	77	87	94	258	
9						
10						

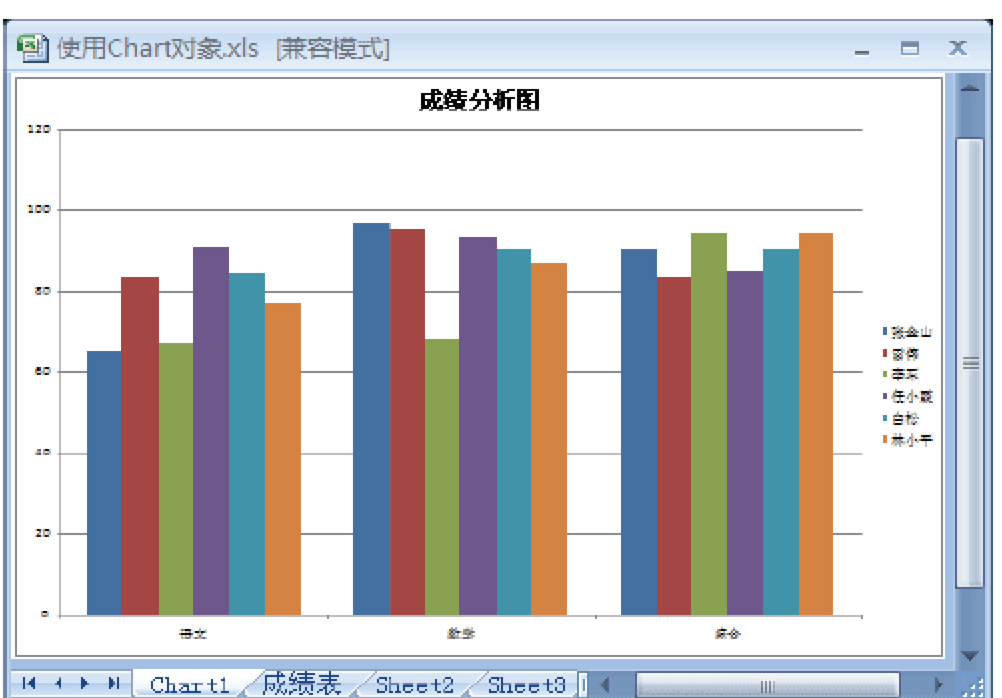


图 15-9　创建图表工作表

15.3.2　创建嵌入图表

图表工作表对象为 Chart 对象，而嵌入到工作表中的图表对象为 ChartObject 对象。ChartObjects 集合包含指定工作表上所有的 ChartObject 对象的集合。

每个 ChartObject 对象都代表一个嵌入图表。ChartObject 对象充当 Chart 对象的容器。ChartObject 对象的属性和方法控制工作表上嵌入图表的外观和大小。

通过 ChartObjects 集合的 Add 方法，可向集合中添加嵌入式图表。其语法格式如下：

```
表达式.Add(Left, Top, Width, Height)
```

该方法的 4 个参数指定嵌入式图表尺寸，分别设置左上角的坐标位置和图表的初始大小。

使用 ChartObjects 集合的 Delete 方法可删除指定工作表的嵌入式图表。

使用以下代码可由工作表“成绩表”中的数据创建一个嵌入式图表：

```
Sub 创建嵌入图表()
    Dim cht As ChartObject
    On Error Resume Next
    ActiveSheet.ChartObjects.Delete        '删除工作表中已有的嵌入图表
    On Error GoTo 0

    With Range("G2:L15")
        Set cht = ActiveSheet.ChartObjects.Add( _
        .Left, .Top, .Width, .Height)       '创建新的嵌入图表
    End With
    With cht
        .Name = "Results"                  '设置嵌入图表的名称
        With .Chart                        '指定数据源
            .SetSourceData Source:=Sheets("成绩表").Range("A2:D8"), PlotBy:
            =xlRows
            .ChartType = xlColumnClustered

            .SetElement msoElementChartTitleCenteredOverlay
```

```
                                            '设置图表标题
            .ChartTitle.Text = "成绩分析图"
        End With
    End With
End Sub
```

以上代码首先删除当前工作表中的嵌入图表，如果当前工作表中没有嵌入图表，则执行 Delete 方法时将出现错误，所以需使用错误捕捉语句获取错误。接着使用 ChartObjects 集合对象的 Add 方法添加一个嵌入式图表，最后设置图表对象的相关属性。

执行以上代码，生成的嵌入图表如图 15-10 所示。

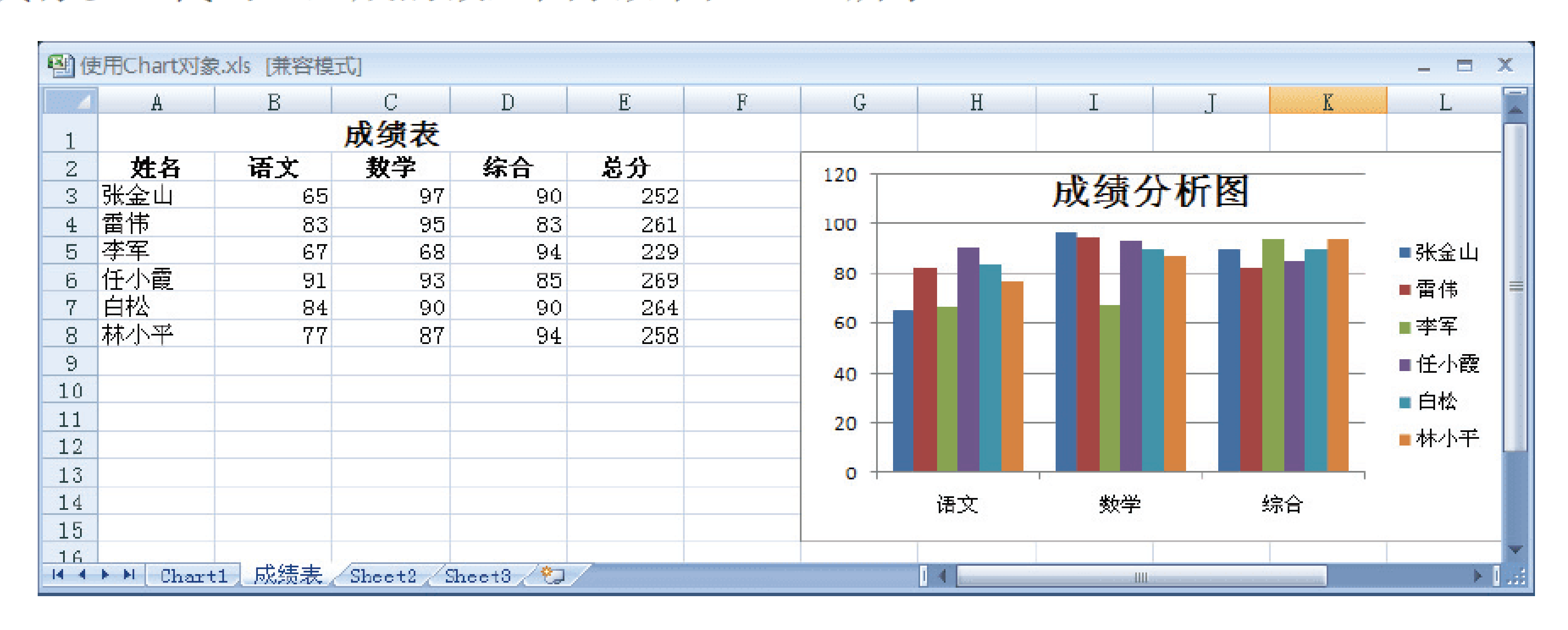


图 15-10　创建嵌入图表

15.3.3　转换图表类型

Excel 中的图表分为两种，即图表工作表和嵌入图表。这两种图表可相互转换。

通过 Chart 对象的 Location 方法，可改变图表的放置位置。该方法的语法格式如下：

```
表达式.Location(Where, Name)
```

两个参数的含义如下所述。

- ❑ Where：用来设置图表移动的目标位置。可设置为 xlLocationAsNewSheet（将图表移动到新工作表）、xlLocationAsObject（将图表嵌入到现有工作表中）或 xlLocationAutomatic（Excel 控制图表位置）3 个常量之一。
- ❑ Name：如果 Where 为 xlLocationAutomatic，则该参数为必选参数。如果 Where 为 xlLocationAsObject，则该参数为嵌入该图表的工作表的名称。如果 Where 为 xlLocationAsNewSheet，则该参数为新工作表的名称。

以下代码可将嵌入图表转为图表工作表：

```
Sub 嵌入图表转换为图表工作表()
    Dim cht As ChartObject
    On Error Resume Next
    Set cht = ActiveSheet.ChartObjects(1)
    If cht Is Nothing Then Exit Sub
    cht.Chart.Location xlLocationAsNewSheet, "成绩分析图"
End Sub
```

以上代码通过工作表的 ChartObjects 集合返回的是一个 ChartObject 对象，要改变其位置，需使用该对象的 Chart 属性返回一个 Chart 对象，通过 Chart 对象的 Location 方法才能改变图表对象的位置。

以下代码可将图表工作表转为嵌入图表：

```
Sub 图表工作表转为嵌入图表()
    Dim cht As Chart, chto As ChartObject
    On Error Resume Next
    Set cht = Charts("成绩分析图")
    If cht Is Nothing Then Exit Sub
    cht.Location xlLocationAsObject, ActiveSheet.Name

    Set chto = ActiveSheet.ChartObjects(1)
    With Range("G2:L15")
        chto.Top = .Top
        chto.Left = .Left
        chto.Width = .Width
        chto.Height = .Height
    End With
End Sub
```

以上代码中，首先获取图表工作表的引用，再通过 Location 方法改变其位置。将图表工作表改为嵌入式图表后，嵌入图表将使用默认的位置。为了不使嵌入图表遮掩数据，程序最后修改了嵌入式图表的位置。

15.3.4　获取图表标题信息

图表对象由多个子对象组成，例如，图表对象包括图表标题对象、图例对象、坐标轴对象等子对象。

如果图表有可见标题，HasTitle 属性为 True。在对图表标题对象进行操作之前，应该先使用该属性判断图表是否有标题。

ChartTitle 对象代表图表标题。使用图表对象的 ChartTitle 属性可返回 ChartTitle 对象。只有图表的 HasTitle 属性为 True 时，ChartTitle 对象才存在，从而才能使用该对象。

通过 ChartTitle 对象的属性和方法可控制图表的标题。例如，使用以下代码可显示图表的标题信息：

```
Sub 图表标题信息()
    Dim chto As ChartObject, cht As Chart
    Dim i As Integer, str1 As String
    i = 1
    For Each chto In ActiveSheet.ChartObjects
        Set cht = chto.Chart
        If cht.HasTitle Then
            With cht.ChartTitle
                str1 = "第" & i & "个嵌入图表的标题信息：" & vbNewLine & vbNewLine
                & _
```

```
                    "名称: " & .Name & vbNewLine & _
                    "文字: " & .Text & vbNewLine & _
                    "字体: " & .Font.Name & vbNewLine & _
                    "字号: " & .Font.Size & vbNewLine & _
                    "颜色: " & .Font.ColorIndex
            End With
        Else
            str1 = "第" & i & "个嵌入图表没有标题! "
        End If
        MsgBox str1
        i = i + 1
    Next
End Sub
```

以上代码将循环处理当前工作表中的各嵌入图表，并分别显示各嵌入图表的标题信息。

15.3.5 图表的系列信息

SeriesCollection 对象表示指定的图表或图表组中所有 Series 对象的集合。使用 Chart 对象的 SeriesCollection 方法可返回 SeriesCollection 集合。

注意：SeriesCollection 对象是一个集合对象。

Series 对象代表图表上的一个系列。Series 对象是 SeriesCollection 集合的成员。使用 SeriesCollection(index)（其中 index 是系列索引号或名称）可返回一个 Series 对象。

在实际使用中，一般不需要明显地定义 Series 对象，而是通过 SeriesCollection(w)的形式访问一个 Series 对象，再通过 Series 对象的 Formula 属性获取以 A1 样式表示系列的公式字符串。

例如，使用以下代码可显示当前图表的系列信息：

```
Sub 系列信息()
    Dim chto As ChartObject, cht As Chart, rng As Range
    Dim i As Integer, j As Integer, w As Integer
    Dim str1 As String, str2 As String, str3 As String
    Dim arr1, arr2
    arr1 = Array("系列", "X 轴数据源", "Y 轴数据源")
    i = 1
    For Each chto In ActiveSheet.ChartObjects
        Set cht = chto.Chart
        str1 = ""
        For w = 1 To cht.SeriesCollection.Count
            str2 = cht.SeriesCollection(w).Formula '获取指定系列的公式
            str2 = Mid(str2, Len("=SERIES(") + 1)  '将公式字符串中前面的字符去掉
            str2 = Left(str2, Len(str2) - 3)       '去掉后面的多余字符

            arr2 = Split(str2, ",")                '将系列各部分分解到数组中
            For j = 0 To UBound(arr2) - 1           '获取各部分的具体值
```

```
                    Set rng = Nothing
                    On Error Resume Next
                    Set rng = Application.Evaluate(arr2(j)) '获取单元格区域
                    On Error GoTo 0
                    If Not rng Is Nothing Then
                        If j = 0 Then
                            str3 = rng.Value                '系列名称字符串
                        Else
                            str3 = rng.Address              '数据源单元格区域地址
                    End If
                    str1 = str1 & arr1(j) & w & ": " & str3 & vbCrLf
                    End If
                Next j
                str1 = str1 & vbNewLine
            Next w
            MsgBox str1                                     '显示一个嵌入图表的系
                                                             列信息
            i = i + 1
        Next
End Sub
```

执行以上代码，将弹出一个对话框如图 15-11 所示，在该对话框中显示了嵌入图表的系列信息。

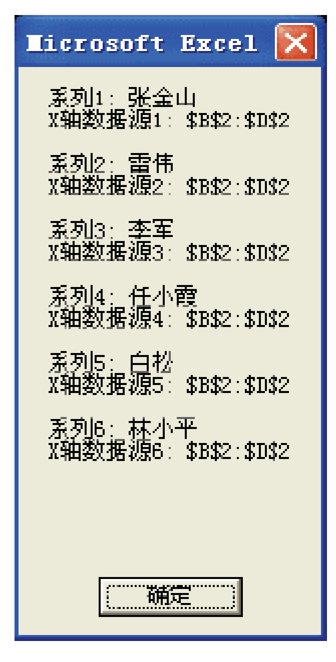


图 15-11　系列信息

15.3.6　调整图表的数据源

使用 Chart 对象的 SetSourceData 方法，可为指定图表设置源数据区域。该方法的语法格式如下：

```
表达式.SetSourceData(Source, PlotBy)
```

两个参数的含义如下所述。

- ❑ Source：指定源数据区域的 Range 对象。
- ❑ PlotBy：设置数据绘制方式。可为常量 xlColumns（数据系列在行中）或 xlRows（数据系列在列中）之一。

使用以下代码可调整图表的数据源：

```
Sub 调整图表数据源()
    Dim myCell As Range
    If ActiveChart Is Nothing Then
        MsgBox "请选择需要调整数据源的图表！"
        Exit Sub
    End If
                                                         '选择需要制作图表的区域
    Set myCell = Application.InputBox(prompt:="请选择调整的数据源区域。",
    Type:=8)

    ActiveChart.SetSourceData Source:=myCell
End Sub
```

以上代码首先判断用户是否选中图表，接着弹出对话框让用户输入或选择新的数据源区域，最后使用 SetSourceData 方法为图表设置新的数据源。

15.3.7 将图表保存为图片

使用 ChartObjects 对象的 CopyPicture 方法，可将选中的图表作为图片复制到剪贴板中。该方法的语法格式如下：

```
表达式.CopyPicture(Appearance, Format)
```

其中，参数 Appearance 表示设置图片的复制方式。可设置为以下两个常量之一。

- ❑ xlScreen：图片尽可能按其屏幕显示进行复制，这是默认值。
- ❑ xlPrinter：图片按其打印效果进行复制。

参数 Format 表示设置图片的格式。可设置为以下两个常量之一。

- ❑ xlBitmap：位图（.bmp、.jpg、.gif）。
- ❑ xlPicture：绘制图片（.png、.wmf、.mix）。

使用以下代码，可将图表保存为图片：

```
Sub 保存为图片()
    If ActiveChart Is Nothing Then
        MsgBox "请选择需要设置格式的图表！"
        Exit Sub
    End If
    ActiveChart.CopyPicture Appearance:=xlScreen, Format:=xlBitmap
    ActiveWindow.Visible = False
    Range("I1").Select
    ActiveSheet.Paste
End Sub
```

15.3.8 使用嵌入图表事件

工作簿中的图表工作表对象 Chart 与工作表对象 Worksheet 具有类似的事件，通过该事件可响应用户对图表的操作。

与图表工作表不同，嵌入图表位于工作表中，因为图表的事件都被工作表截获，从而不能对图表实现事件驱动。要捕获嵌入图表的事件，还需要通过一个类模块来进行转换，具体步骤如下：

（1）创建一个类模块，在其中声明一个公共的 Chart 对象，声明该对象时必须使用 WithEvents 关键字，该关键字说明声明的变量是用来响应由 ActiveX 对象触发的事件的对象变量。只有在类模块中声明才是合法的，其代码如下：

```
Public WithEvents myChartClass As Chart
```

（2）在类模块中定义需要捕获的事件，并编写相应的事件过程代码。例如，要捕获 Select 事件，可以编写以下事件过程：

```
Private Sub myChartClass_Select(ByVal ElementID As Long,  _
   ByVal Arg1 As Long, ByVal Arg2 As Long)
```

（3）创建一个模块，用前面定义的类模块声明一个变量，例如以下的代码：

```
Dim MyChart As New myChartClass
```

（4）在模块中编写代码，将变量与某个工作表中的嵌入图表建立起连接，例如以下的代码：

```
Sub EnableChartClass()
    Set MyChart.myChartClass = Worksheets("成绩表").ChartObjects(1).Chart
End Sub
```

下面以具体的实例演示激活嵌入式图表事件的方法，具体步骤如下：

（1）在有嵌入图表的工作簿中进入 VBE。

（2）在 VBE 中单击主菜单【插入】|【类模块】命令，插入一个类模块。

（3）在属性窗口中修改类模块的名称为 myChartClass，如图 15-12 所示。



图 15-12　定义类模块名称

（4）在类模块中输入以下代码定义一个公共变量。

```
Public WithEvents myChartClass As Chart
```

（5）在类模块 myChartClass 中编写事件过程代码。在类模块 myChartClass 的对象下拉列表中选择 myChartClass，在事件下拉列表中选择 Select，系统将自动生成事件过程的结构，并自动填入事件的参数。具体的代码如下：

```
Private Sub myChartClass_Select(ByVal ElementID As Long, ByVal Arg1 As Long,
```

```
ByVal Arg2 As Long)
    Dim str1 As String
    Select Case ElementID
        Case xlChartArea: str1 = "图表区"
        Case xlChartTitle: str1 = "图表标题"
        Case xlPlotArea: str1 = "绘图区"
        Case xlLegend: str1 = "图例"
        Case xlLegendEntry: str1 = "图例项"
        Case xlLegendKey: str1 = "图例标示"
        Case xlAxis: str1 = "坐标轴"
        Case xlAxisTitle: str1 = "坐标轴标题"
        Case xlMajorGrstr1lines: str1 = "主要网格线"
        Case xlMinorGrstr1lines: str1 = "次要网格线"
        Case xlDataLabel: str1 = "数据标签"
        Case xlDataTable: str1 = "数据表"
        Case xlDropLines: str1 = "垂直线"
        Case xlErrorBars: str1 = "误差线"
        Case xlHiLoLines: str1 = "高低点连线"
        Case xlSeries: str1 = "系列"
        Case xlSeriesLines: str1 = "系列线"
        Case xlShape: str1 = "图形"
        Case xlFloor: str1 = "基底"
        Case xlWalls: str1 = "背景墙"
         Case xlNothing: str1 = "Nothing"
       Case Else:: str1 = "未识别对象"
    End Select
    MsgBox "你选择的是:  " & str1
End Sub
```

（6）在 VBE 中单击主菜单【插入】|【模块】命令，插入一个名称为“模块 2”的模块，在模块的声明部分定义一个模块变量。

```
Dim MyChart As New myChartClass
```

（7）在“模块 2”中编写以下代码，将模块中的变量与“成绩表”中的嵌入图表连接起来。

```
Sub EnableChartClass()
    Set MyChart.myChartClass = Worksheets("成绩表").ChartObjects(1).Chart
End Sub
```

执行过程 EnableChartClass 后，在嵌入图表上单击时，嵌入式图表将响应用户的操作事件，执行类模块中的 myChartClass_Select 事件过程，并弹出一个对话框，显示单击处的子对象名称。

（8）如果要禁止嵌入图表响应用户事件，可设置对象变量 MyChart 的值为 Nothing。代码如下：

```
Sub DisableChartClass()
    Set MyChart = Nothing
End Sub
```

第 4 部分　用户界面设计

在 Excel 中，用户大部分时间是在工作表中进行操作。在使用 VBA 开发应用程序时，为了保护数据、简化用户操作，可通过设计美观的用户窗体，使用户直接在窗体上进行操作即可将数据保存到工作表中。

和以前版本的 Excel 相比，Excel 2007 采用了全新的面向结果的用户界面。以前版本中熟悉的菜单栏和工具栏消失了，被称为“功能区”（Ribbon）的面板取代。

本部分共 6 章，详细介绍了在 Excel 中开发应用程序的用户界面设计知识。

第 16 章　使用 Excel 内置对话框

Excel 使用了大量的内置对话框，例如，新建、打开、保存工作簿，设置单元格格式等。在 VBA 中，Excel 提供了调用这些内置对话框的接口，使开发者可充分利用内置对话框实现更多操作。本章将介绍在 VBA 中调用 Excel 内置对话框的方法。

16.1　了解 Excel 内置对话框

在 VBA 中，Application 对象提供了许多方法，用来显示信息或接收用户输入。Application 对象提供以下方法，可以打开相应的内置对话框。

- FindFile 方法：显示【打开】对话框。
- GetOpenFilename 方法：显示标准的【打开】对话框，并获取用户文件名，而不必真正打开任何文件。
- GetSaveAsFilename 方法：显示标准的【另存为】对话框，获取用户文件名，而无须真正保存任何文件。
- InputBox 方法：显示一个接收用户输入的对话框。返回此对话框中输入的信息。

在本书第 5 章中曾介绍了 InputBox 方法、MsgBox 函数的使用方法。本章将介绍其他方法打开的对话框。

更多的内置对话框是由 Application 对象的 Dialogs 属性提供，该属性返回一个 Dialogs 集合，该集合包含了 Excel 所有的内置对话框。本章第 5 节将介绍使用 Dialogs 属性调用内置对话框的方法。

16.2　使用 FindFile 打开文件

使用 Application 对象的 FindFile 方法，可显示如图 16-1 所示的【打开】对话框，在该对话框中选择一个文件，单击【打开】按钮打开一个文件。如果成功打开一个新文件，则该方法返回 True。如果用户单击【取消】按钮退出该对话框，则该方法返回 False。

在【打开】对话框的【文件类型】下拉列表框中，列出了 Excel 能打开的各种文件类型。

使用以下代码可显示【打开】对话框：

```
Sub 打开工作簿()
```

```
    Dim wb As Workbook
    Set wb = ActiveWorkbook
    If Application.FindFile Then
        MsgBox "已打开工作簿文件！"
        wb.Activate
    Else
        MsgBox "打开工作簿失败！"
    End If
End Sub
```

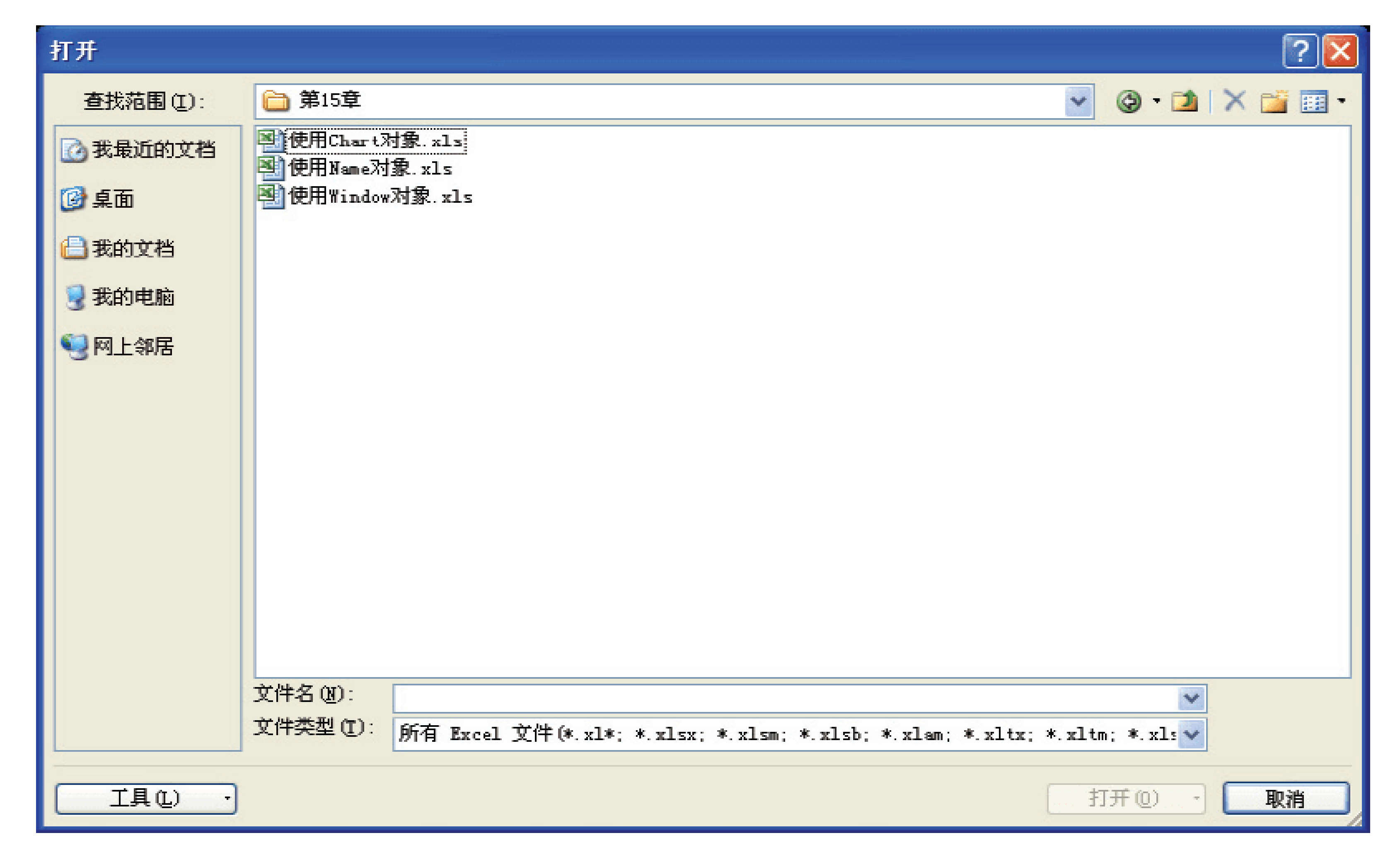


图 16-1　【打开】对话框

16.3　使用 GetOpenFilename 获取文件名

使用 Application 对象的 FindFile 方法可显示【打开】对话框，用户在对话框中选择文件名后，单击【打开】按钮将打开选中的文件。在某些情况下，可能程序首先需要获取用户要打开的一个或多个文件名，经过 VBA 代码进行处理后，再调用相应的命令打开文件。这时，可使用 Application 对象的 GetOpenFilename 方法。

16.3.1　GetOpenFilename 方法

使用 Application 对象的 GetOpenFilename 方法，将打开一个标准的【打开】对话框，让用户在计算机中选择盘符、路径、文件类型和文件名等信息。当用户在该对话框中单击【打开】按钮时，将返回选择的路径和文件名，但并不真正执行打开操作。其语法格式如下：

```
表达式.GetOpenFilename(FileFilter, FilterIndex, Title, ButtonText, Multi
```

```
Select)
```

该表达式中的所有参数都是可选的，各参数的描述如下所述。

- FileFilter：一个指定文件筛选条件的字符串。

在 FileFilter 参数中传递的字符串由文件筛选字符串以及后跟的 DOS 通配符文件筛选规范组成，中间以逗号分隔。每个字符串都在“文件类型”下拉列表框中列出。例如，下列字符串指定两个文件筛选——文本和加载宏：

```
“文本文件 (*.txt)、*.txt、加载宏文件 (*.xla)、*.xla”。
```

如果省略 FileFilter，则此参数默认为：

```
“所有文件 (*.*),*.*”
```

要为单个文件筛选类型使用多个通配符表达式，需用分号将通配符表达式分开。例如：

```
"Excel 文件(*.xls;*.xlsx;*.xlsm),*.xls;*.xlsx;*.xlsm,"
```

- FilterIndex：指定默认文件筛选条件的索引号，取值范围为 1 到 FileFilter 所指定的筛选条件数目。如果省略该参数，或者该参数的值大于可用筛选条件数，则使用第一个文件筛选条件。
- Title：指定对话框的标题。如果省略该参数，则标题为“打开”。
- ButtonText：用于设置对话框中按钮上的文本，在 PC 机中不能使用。
- MultiSelect：如果为 False（默认值），则只允许选择一个文件名。如果为 True，则允许选择多个文件名，返回值是一个包含所有选定文件名的数组（即使仅选定了一个文件名）。

16.3.2　获取单个文件名

使用 Application 对象的 GetOpenFilename 方法时，MultiSelect 参数的默认值为 False，表示只允许用户在对话框中选择一个文件名。例如，下面的代码提示用户选择文件名，在这段代码中，用户一次只能选择一个文件名。

```
Sub 获取单个文件名()
   Dim strFilt As String
   Dim strTitle As String
   Dim strFname As Variant
   strFilt = "文本文件(*.txt),*.txt," & _
          "Excel 文件(*.xls;*.xlsx;*.xlsm),*.xls;*.xlsx;*.xlsm," & _
          "所有文件(*.*),*.*"
   strTitle = "打开 Excel 文件"
   strFname = Application.GetOpenFilename _
       (filefilter:=strFilt, _
       Title:=strTitle)
   If strFname = False Then
```

```
        MsgBox "没选择文件! "
    Else
        MsgBox "选择的文件是: " & strFname
    End If
End Sub
```

注意：接受 GetOpenFilename 方法返回值的变量必须是一个 Variant。

执行以上代码，在显示标准的【打开】对话框中，用户可改变查找范围、更换文件类型，选中需要打开的文件后单击【打开】按钮，将返回文件的路径和名称，供打开程序的代码使用。

16.3.3　获取多个文件名

如果允许用户一次选择多个文件，则需要将 GetOpenFilename 方法的 MultiSelect 参数设置为 True。例如，下面的代码将允许用户在【打开】对话框中选择多个文件。

```
Sub 获取多个文件名()
    Dim strFilt As String
    Dim strTitle As String
    Dim strFname As Variant
    Dim i As Integer
    Dim strMsg As String
    strFilt = "文本文件(*.txt),*.txt," & _
            "Excel 文件(*.xls;*.xlsx;*.xlsm),*.xls;*.xlsx;*.xlsm," & _
            "所有文件(*.*),*.*"
    strTitle = "打开 Excel 文件"
    strFname = Application.GetOpenFilename _
        (filefilter:=strFilt, _
        Title:=strTitle, _
        MultiSelect:=True)            '允许选择多个文件
    If Not IsArray(strFname) Then     '返回值不是数组
        MsgBox "没选择文件! "
    Else
        For i = LBound(strFname) To UBound(strFname)
            strMsg = strMsg & strFname(i) & vbCrLf
        Next
        MsgBox "选择的文件是: " & vbNewLine & strMsg
    End If
End Sub
```

执行以上代码，将显示如图 16-2 所示的【打开】对话框，按住 Ctrl 键，并在对话框中单击选中需要打开的多个文件，然后单击【打开】按钮，选中的文件名将返回到一个数组

中，最后使用 MsgBox 函数将用户选中的文件名显示出来，如图 16-3 所示。

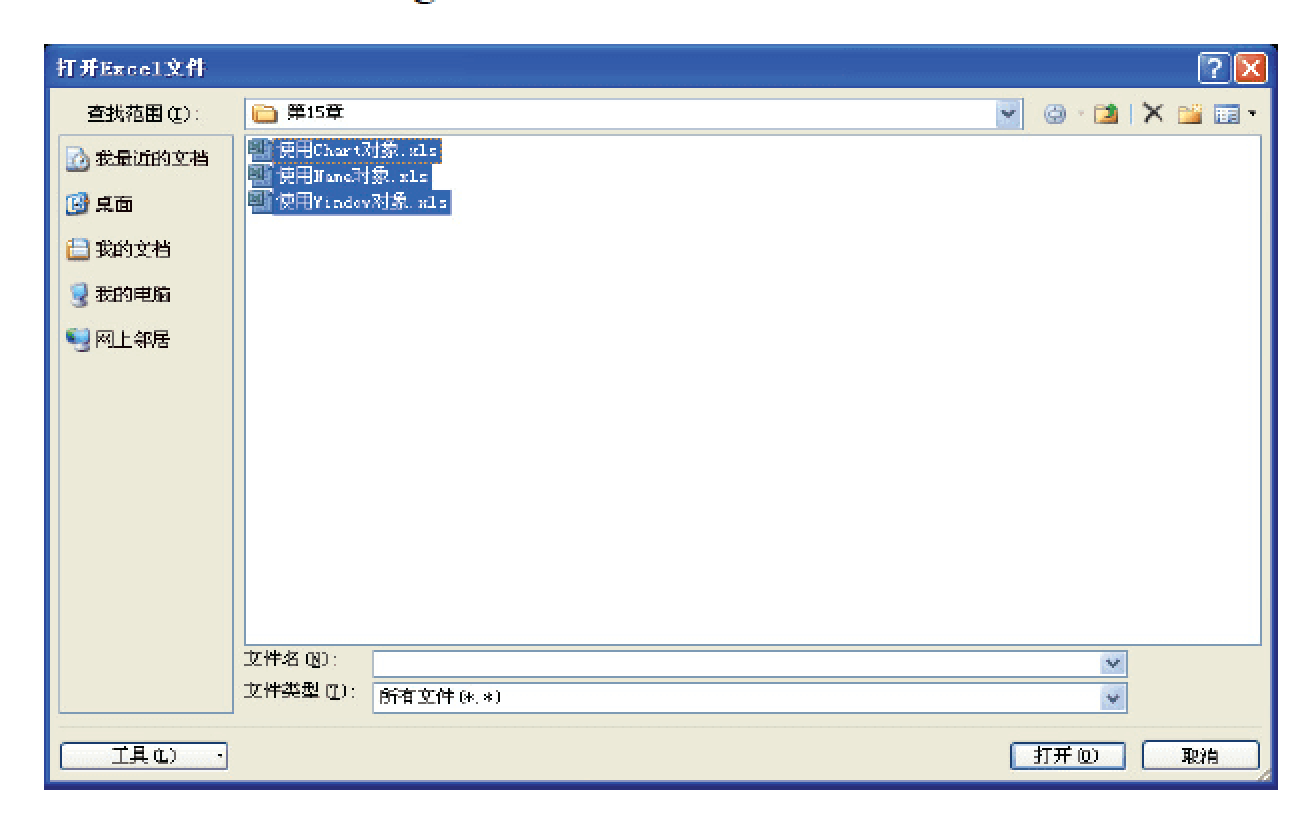



图 16-2 选择多个文件名

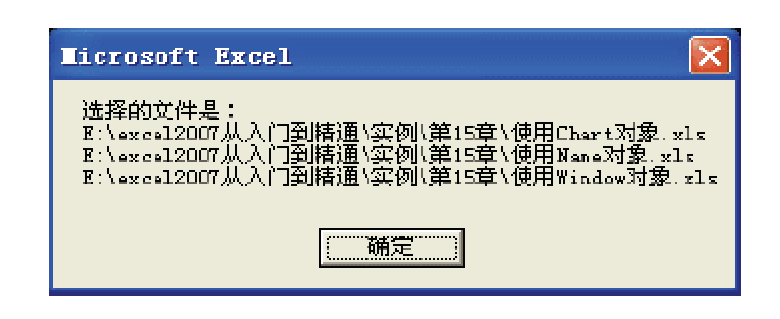


图 16-3 选择的文件

16.4 使用 GetSaveAsFilename 获取保存文件名

与 GetOpenFilename 方法类似，使用 Application 对象的 GetSaveAsFilename 方法可打开标准的【另存为】对话框，在该对话框中用户可以选择（或输入）一个文件名。其语法格式如下：

```
表达式.GetSaveAsFilename(InitialFilename, FileFilter, FilterIndex, Title, ButtonText)
```

其参数与 GetOpenFilename 方法类似，这里不再详述。

注意：GetOpenFilename 方法只返回文件名及其路径，而不执行具体的保存操作。

下面的代码将打开一个【另存为】对话框：

```
Sub 获取保存文件名()
    Dim fn As String
    fn = Application.GetSaveAsFilename( _
        fileFilter:="Text Files (*.txt), *.txt")
    If fn <> "False" Then
        '在此处编写保存文件的代码
        MsgBox "保存文件为: " & fn
    End If
End Sub
```

执行以上代码，将显示如图 16-4 所示的【另存为】对话框，选择好保存路径后，输入保存文件名，单击【保存】按钮将关闭该对话框，并返回文件名及其位置，在随后的代码中可将文件保存在设置的文件名中。

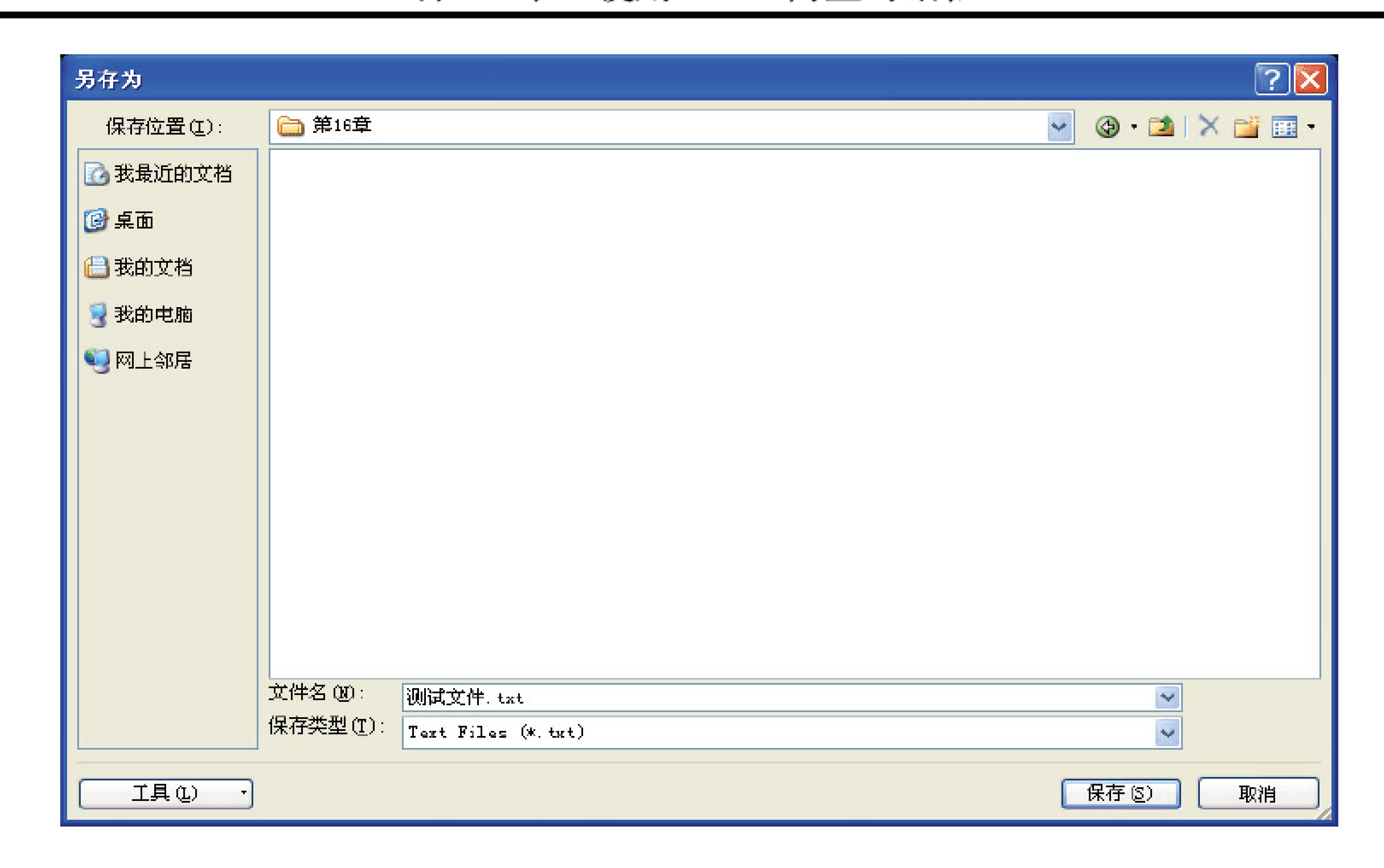


图 16-4 【另存为】对话框

16.5　调用 Excel 内置对话框

Application 对象除了通过前面介绍的几个方法显示常用的内置对话框外，还可使用 Dialogs 集合访问所有的 Excel 内置对话框。

16.5.1　Dialogs 集合和 Dialog 对象

使用 Appliation 对象的 Dialogs 属性返回一个 Dialogs 集合，该集合包含了 Excel 所有的内置对话框。

1. Dialogs集合

Dialogs 集合为 Excel 中所有 Dialog 对象的集合。每个 Dialog 对象代表一个内置对话框。Dialogs 集没有提供任何方法，因此不能向该集合中新建或添加内置对话框。该集合常用的属性为 Count 和 Item。

使用以下代码可显示 Excel 内置对话框的数量：

```
MsgBox Application.Dialogs.Count
```

在 Excel 2007 中，以上代码将显示 1101，即 Excel 2007 有 1101 个内置对话框。在 Excel 2003 中运行以上代码，将显示为 849，即 Excel 2003 中有 849 个内置对话框。

2. Dialog对象

Dialog 对象代表内置的 Excel 对话框。可使用 Dialogs 集合中的 Item 属性访问某个指定的内置对话框。

Dialog 对象使用 Show 方法显示内置的对话框，等待用户输入数据，然后返回一个代表用户响应的 Boolean 值。其语法格式如下：

```
表达式.Show(Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9, Arg10, Arg11,
Arg12, Arg13, Arg14, Arg15, Arg16, Arg17, Arg18, Arg19, Arg20, Arg21, Arg22,
Arg23, Arg24, Arg25, Arg26, Arg27, Arg28, Arg29, Arg30)
```

Show 方法具有 30 个可选参数，分别为 Arg1～Arg30，这些参数可为内置对话框设置初始参数。

Show 方法的返回值为一个 Boolean 值。如果用户在内置对话框中单击【确定】按钮，则 Show 方法的返回值为 True；如果用户单击【取消】按钮，则返回值为 False。

对于一些内置对话框（例如【打开】对话框），可以使用参数 arg1、arg2、...、arg30 设置初始值。

例如，使用以下代码，可显示【打开】对话框：

```
b = Application.Dialogs(1).Show
```

对于上千个内置对话框，如果使用数值代码表示，将很不直观，也不方便代码的阅读。VBA 中为每个对话框设置了一个常数，通过常数的英文字母组合可使代码更易阅读。

例如，使用 xlDialogOpen 表示【打开】对话框，以下代码也可显示【打开】对话框：

```
b = Application.Dialogs(xlDialogOpen).Show
```

Excel 的内置对话框非常多，要获得所有对话框常量，可以查询帮助系统的 XlBuiltInDialog 枚举，如图 16-5 所示。

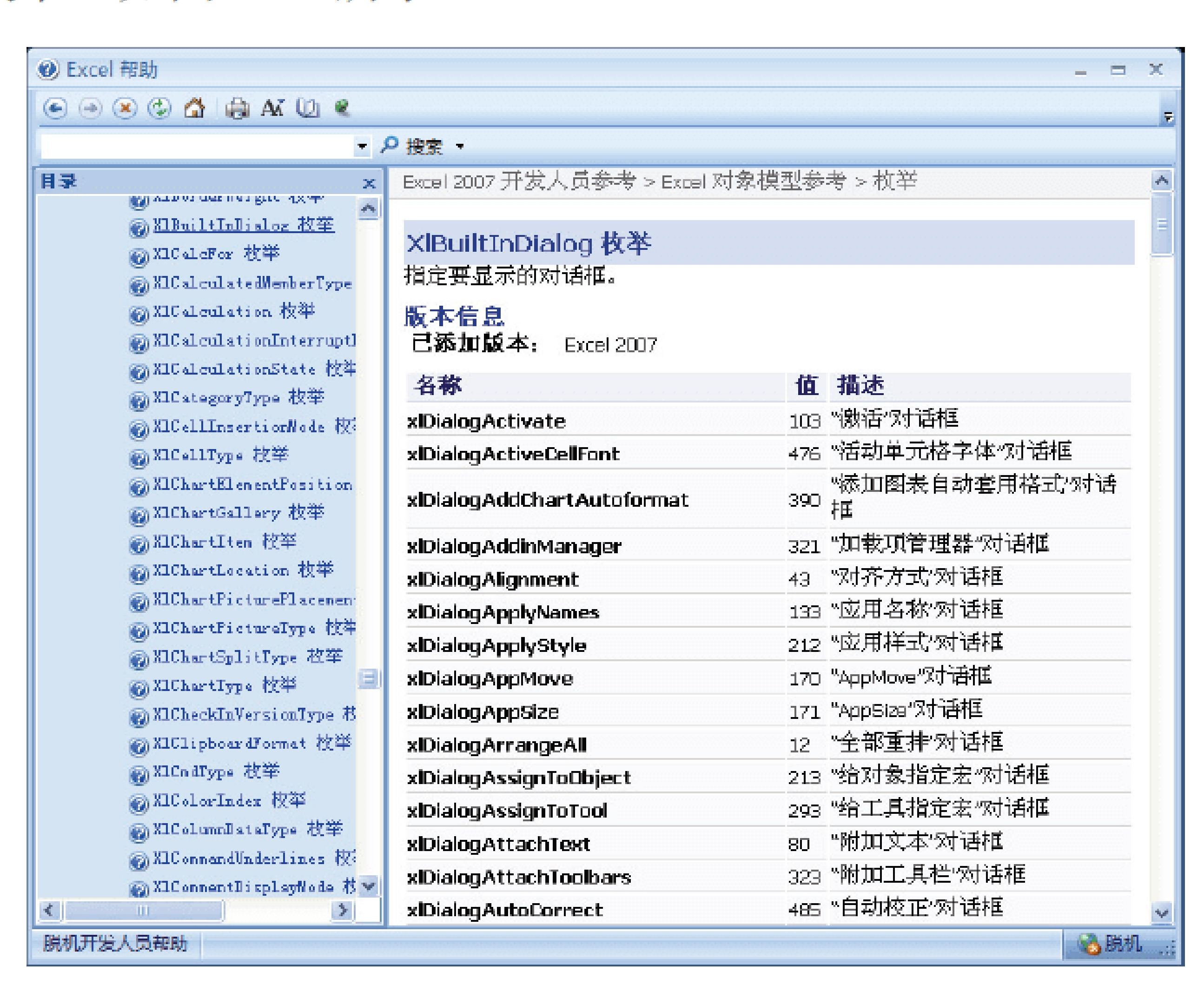


图 16-5 XlBuiltInDialog 枚举

在 VBA 中，Excel 将多页对话框当作多个对话框。例如，在 Excel 中显示的【设置单元格格式】对话框由多个页组成，在 VBA 中就无法将这种由多个页组成的对话框显示出来，但可以分别显示【数字】、【对齐】、【字体】、【边框】、【填充】和【保护】6

个标签页面，分别使用 6 个常量：xlDialogFormatNumber，xlDialogAlignment，xlDialogActiveCellFont，xlDialogBorder，xlDialogColorPalette 和 xlDialogCellProtection 来代表这 6 个标签页面。

以下代码就是显示【对齐】标签页的效果。

```
Application.Dialogs(xlDialogAlignment).Show
```

执行以上代码，将在【设置单元格格式】对话框中显示【对齐】标签页的内容，如图 16-6 所示。

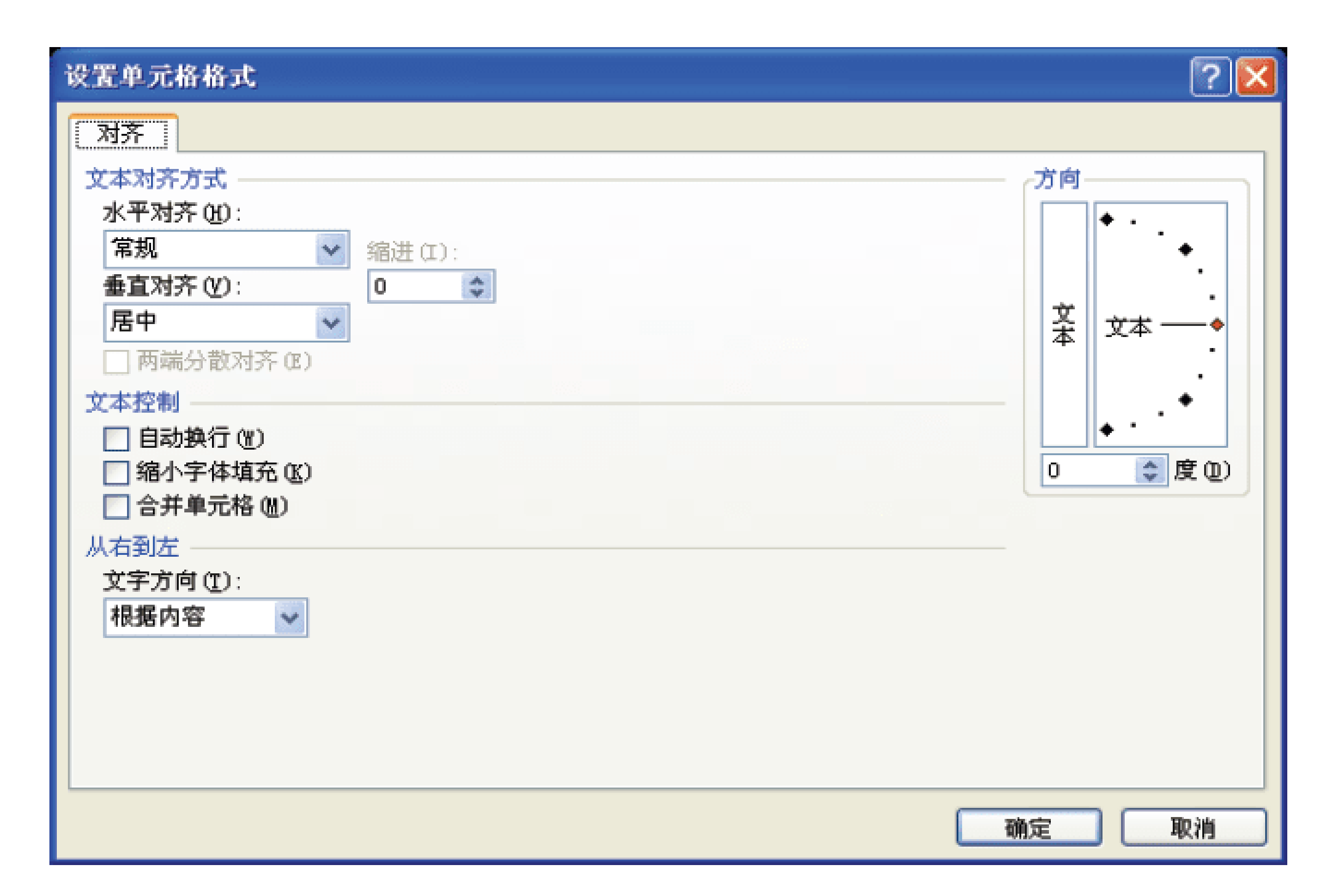


图 16-6 【对齐】标签页

16.5.2　使用内置对话框的初始值

在一般情况下，使用 Dialog 对象的 Show 方法显示 Excel 内置对话框时，对话框中的各控件都使用默认的初始值。

大多数内部 Excel 对话框都有很多参数，可用于设置和取得对话框中的值。为了设置内置对话框选项，可使用相应的参数。

例如，使用以下代码可显示【设置单元格格式】对话框的【字体】标签页，如图 16-7 所示。

```
Application.Dialogs(xlDialogFontProperties).Show
```

打开该对话框后，字体、字形、字号等都使用默认的值。在 VBA 程序中，若需要打开该对话框时，使字体、字形、字号等使用另外一个不同的值，则可在 Show 方法中输入不同的参数，使对话框在显示出来时就具有不同的初始值。

通过 16.5.1 节介绍的 Show 方法的语法格式可知道，Show 方法最多可使用 30 个参数。不同的内置对话框所使用的参数个数也不相同。在【帮助】窗口中搜索“内置对话框参数列表”，结果如图 16-8 所示。

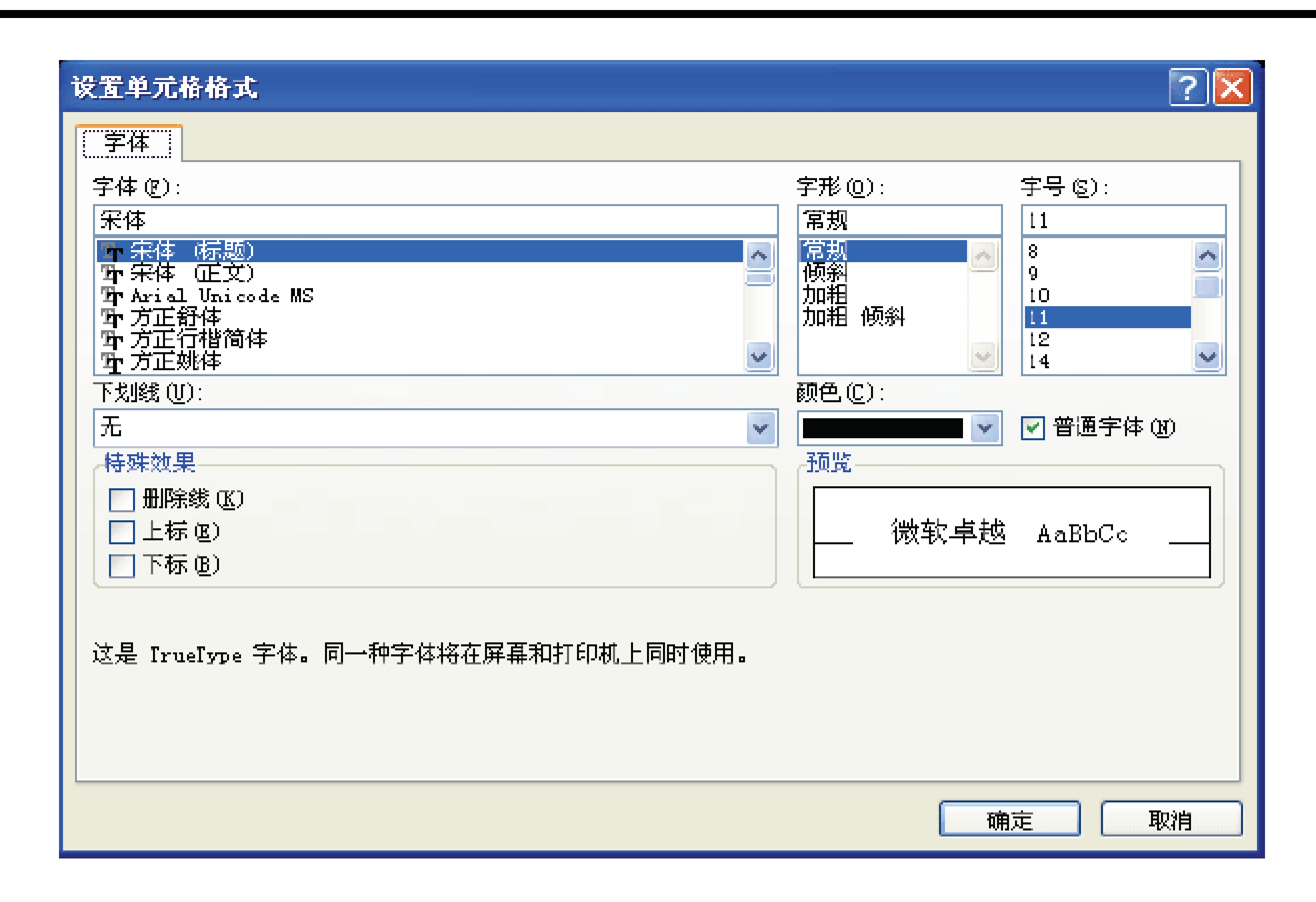


图 16-7 【字体】标签页

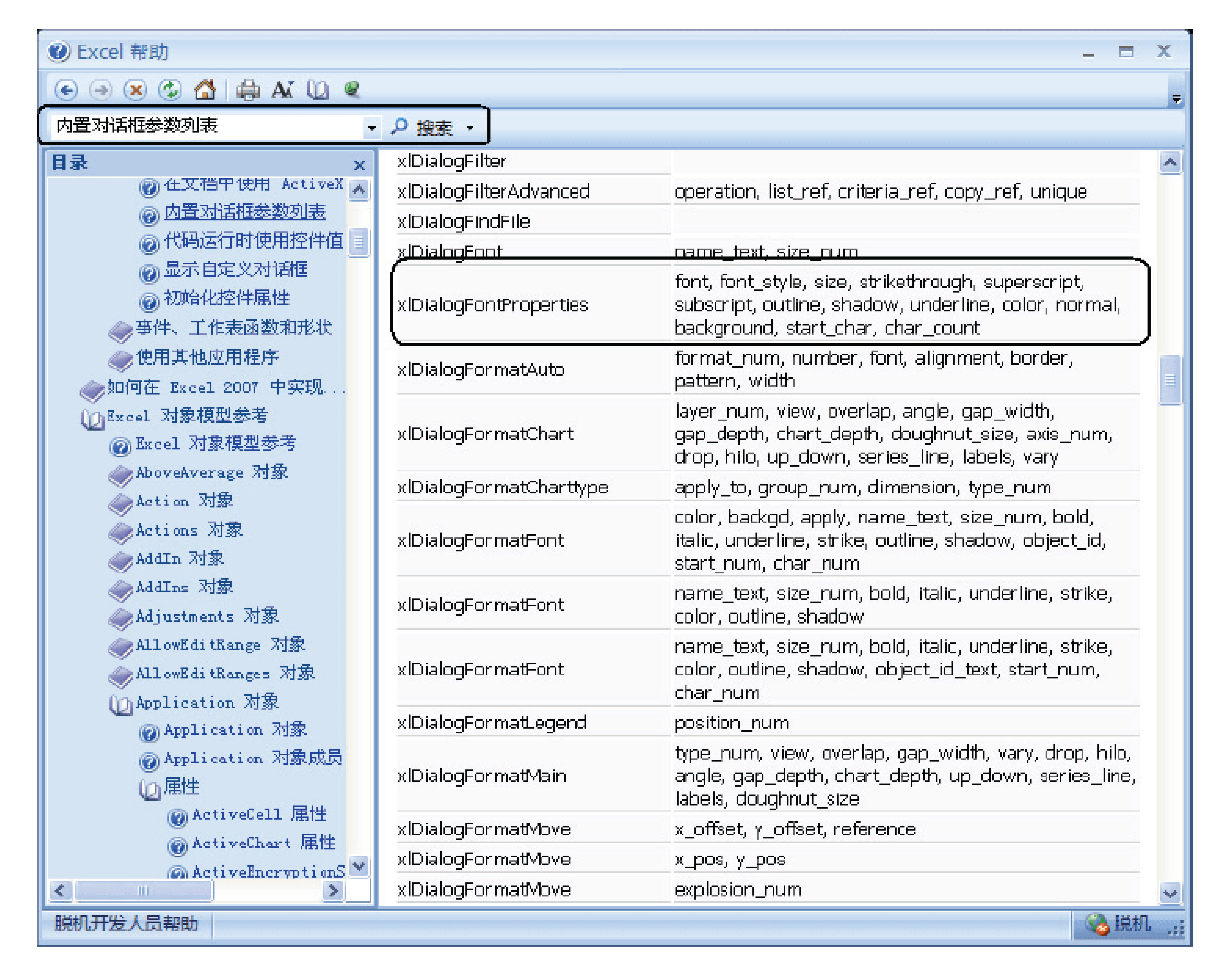


图 16-8 内置对话框参数列表

在图 16-8 中，通过对话框常量可找到相应对话框可以使用的参数名称。例如，可查看到 xlDialogFontProperties 对话框的参数如下：

font, font_style, size, strikethrough, superscript, subscript, outline, shadow, underline, color, normal, background, start_char, char_count。

以上共有 14 个参数。在【字体】对话框中，可使用以下几个参数设置对话框的初始值。

- ❑ font：字体，以文字形式表示要设置的初始字体，例如黑体；
- ❑ font_style：字形，以文字形式表示要设置的字形，例如粗体；

- size：字号，以阿拉伯数字表示字的大小，例如 20；
- strikethrough：删除线，设置为 True 或 False；
- superscript：上标，设置为 True 或 False；
- subscript：下标，设置为 True 或 False；
- underline：下划线，可设置数字 1～5，分别表示下划线在列表框中的位置；
- color：颜色，设置为数字表示颜色索引值。

了解以上参数的含义后，可使用以下方式调用【字体】对话框：

```
Application.Dialogs(xlDialogFontProperties).Show _
    "黑体", "粗体", 20, True, True, False, False, True, 5, 4, True, 3
```

执行以上代码，打开的对话框如图 16-9 所示。

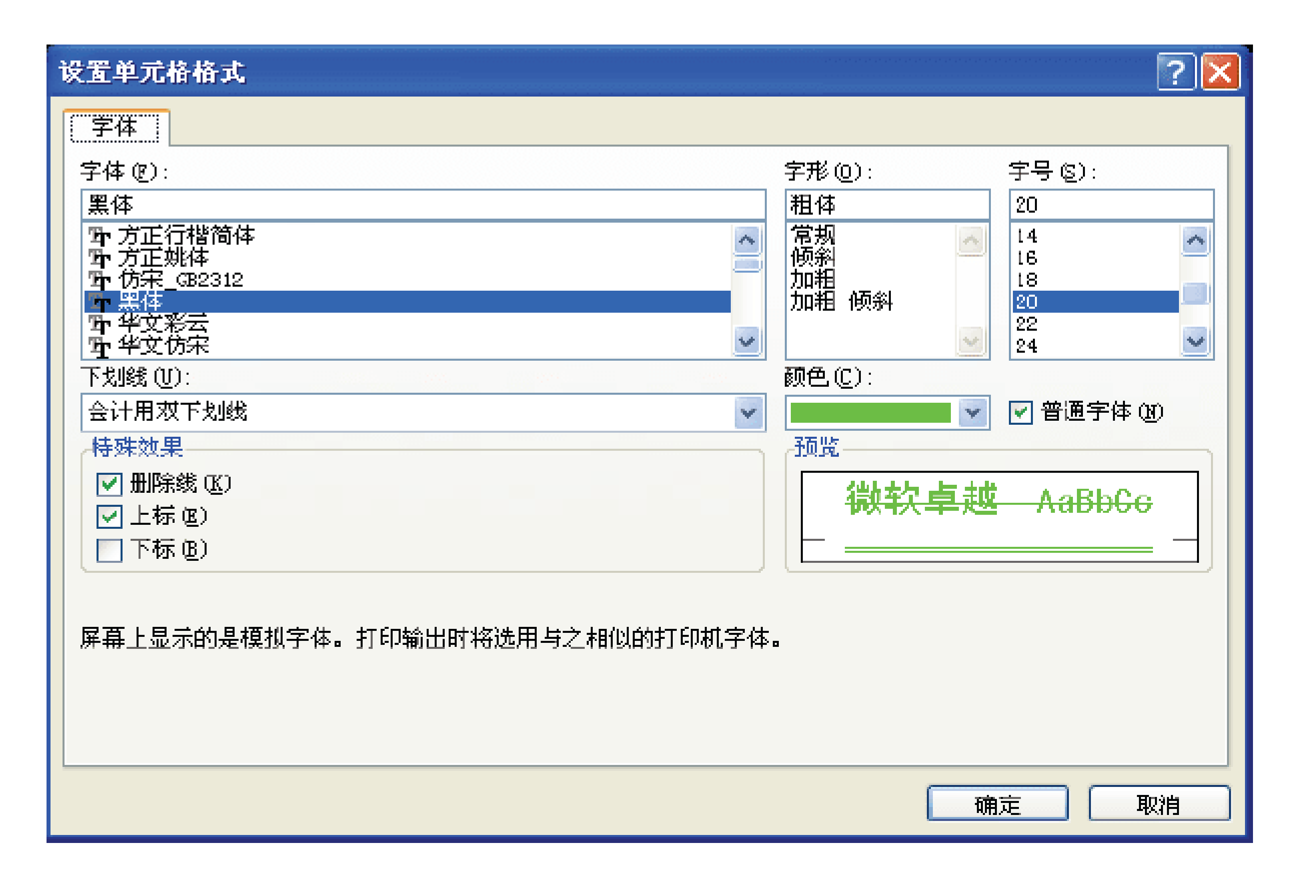


图 16-9　设置初始值的对话框

在以上代码中，各参数的顺序必须按照【帮助】窗口里列出的顺序给出。不用的参数也需要用一个逗号（，）分隔。Show 方法的参数也支持命名参数，使用命名参数的方式可不按顺序给出各参数，代码可改写为以下形式：

```
Application.Dialogs(xlDialogFontProperties).Show _
    arg1:="黑体", arg2:="粗体", arg3:=20, _
    arg4:=True, arg5:=True, arg7:=True, arg8:=True, _
    arg9:=5, arg10:=4, arg11:=True, arg12:=3
```

注意： 参数不是使用帮助系统中列出的 font, font_style, size 之类的关键字，而是使用与参数出现顺序对应的 Arg1、Arg2 等。例如，参数 arg1 对应 font，参数 arg2 对应 font_style。xlDialogFontProperties 对话框共有 14 个参数，分别用 arg1~arg14 与各参数对应。

第 17 章　创建自定义对话框

在 VBA 中，用户可创建自定义对话框，在对话框中放置控件，可接收用户输入数据，也可将应用程序的数据显示在该对话框中。本章将介绍创建自定义对话框的方法。

17.1　新 建 窗 体

在第 16 章中介绍了调用 Excel 内置对话框的方法。这些内置对话框只能使用固定格式完成相应的功能。在很多情况下，需要开发人员创建自定义对话框，用来接受用户输入数据，或显示信息系统中的数据。本节以创建一个【登录】窗体为例，介绍创建用户窗体的过程。

17.1.1　新建窗体

要创建自定义对话框，需要在 VBE 中插入一个用户窗体，再新建窗体，具体步骤如下。

（1）在 Excel 环境中，找到【开发工具】功能区的【代码】组，单击 Visual Basic 按钮，打开 VB 编辑器。

提示：也可以按键盘上的快捷键 Alt+F11 打开 VB 编辑器。

（2）在 VBE 中单击主菜单的【插入】|【用户窗体】命令，向工程中插入一个用户窗体，如图 17-1 所示。

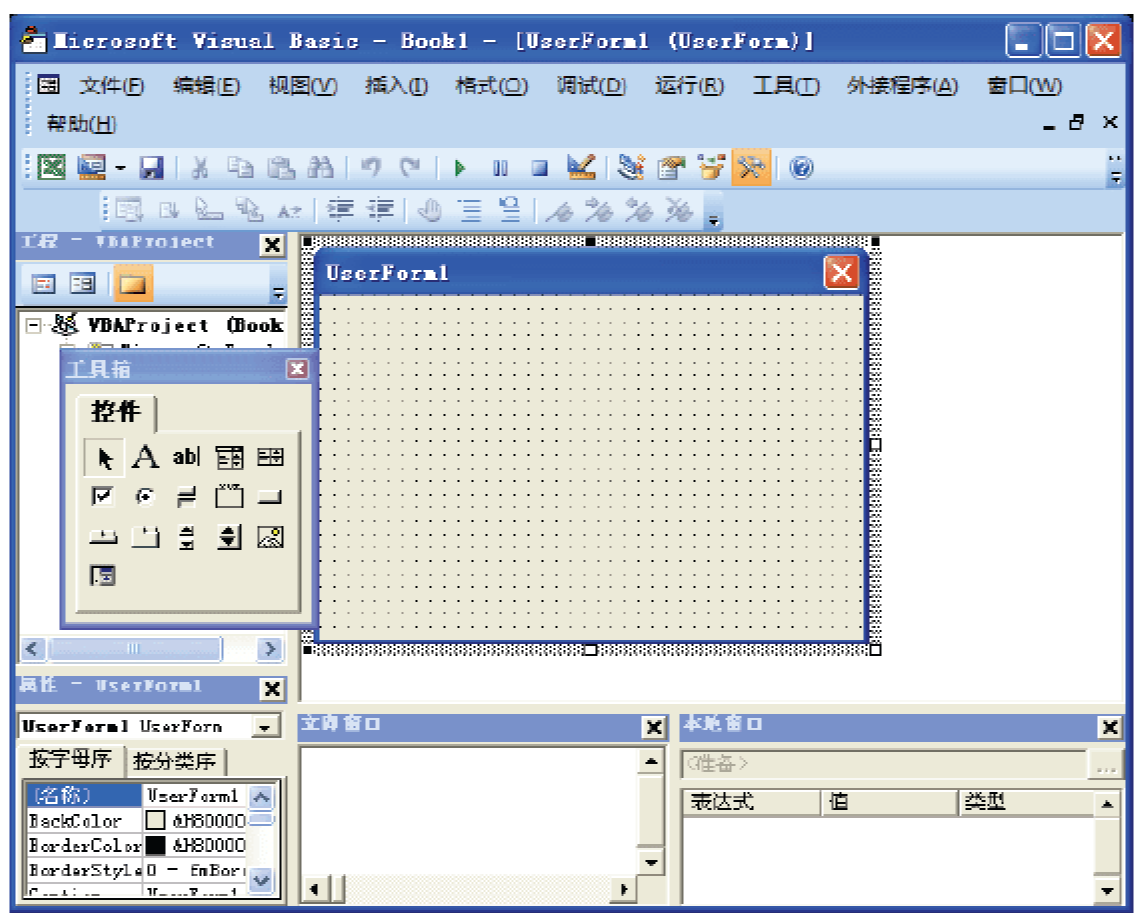


图 17-1　创建窗体

在图 17-1 所示的 VBE 窗口中，右侧主窗体中显示了一个名为 UserForm1 的用户窗体。当显示用户窗体时，在 VBE 窗口中将显示一个【工具箱】子窗体，该子窗体为悬浮状，用户可根据需要将其拖到其他位置。

通过 VBE【工程】资源管理器窗口中的快捷菜单也可向工程中插入用户窗体，如图 17-2 所示。从弹出的快捷菜单中依次选择【插入】|【用户窗体】命令，即可创建一个窗体。

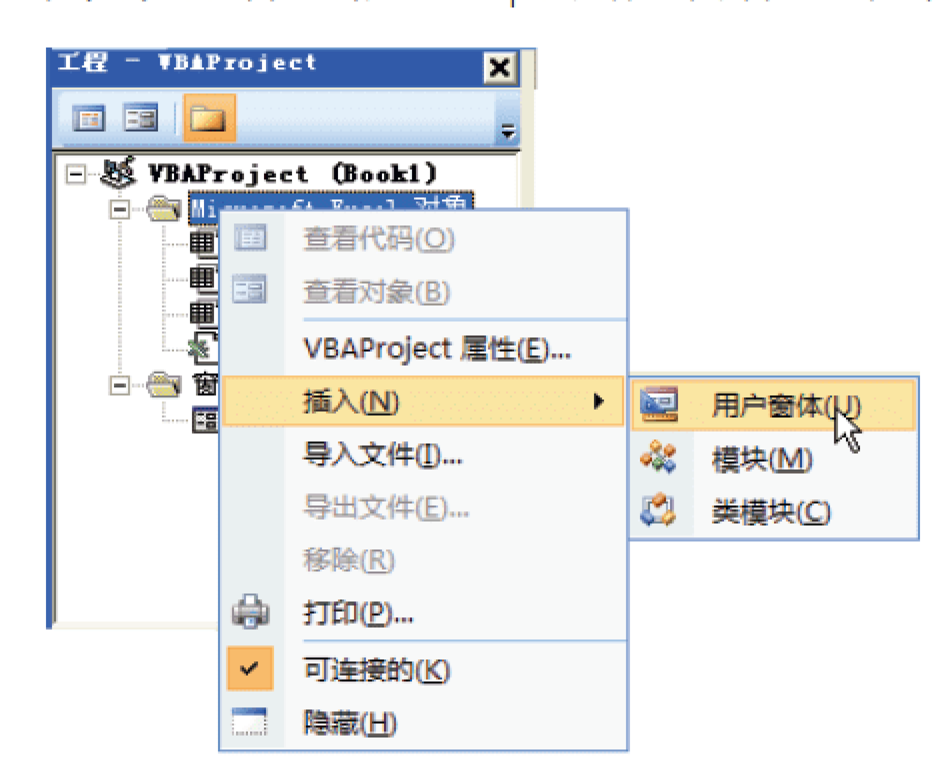


图 17-2　通过快捷菜单插入用户窗体

17.1.2　设置窗体属性

用户窗体作为 VBA 中的一个对象，也可以通过属性、方法和事件对其进行控制。对窗体的事件过程编写代码将在第 17.5 节中进行介绍。下面介绍用户窗体的常用属性。

创建用户窗体后，窗体的属性都是默认值。开发人员可以通过【属性】窗口修改属性值。窗体的常用属性如下所述。

- 名称（Name）属性：为窗体设置一个名称，在程序运行时，该名称代表窗体。在程序运行时窗体名称不允许修改。用户窗体的默认名称是 UserForm 加上一个整数，例如，第一个用户窗体为 UserForm1，第二个为 UserForm2。
- BackColor 属性和 ForeColor 属性：BackColor 属性设置窗体的背景颜色，ForeColor 设置窗体的前景色（即窗体中显示的文本和图形的颜色）。
- BorderStyle 属性：通过该属性为窗体设置边框样式，在 VBA 中窗体共有两种边框，分别为两个常量：
  - fmBorderStyleNone，值为 0，表示窗体无可见的边框线。
  - fmBorderStyleSingle，值为 1，表示控件有一单线的边框，这是默认值。
- Caption 属性：通过该属性设置窗体标题栏显示的文字。当窗体最小化时，该文字显示在窗体任务栏中。创建窗体时，Caption 属性的值是 UserForm 加上一个整数。
- Enable 属性：通过该属性设置窗体是否对用户的操作进行响应，其有 True 和 False 两个值，如果设置窗体的 Enable 属性为 False，则该窗体将不能被移动，也不能调整大小，窗体上的控件也不响应用户的操作。
- Height 属性和 Width 属性：这两个属性设置窗体的高度和宽度，以磅为单位。当

改变窗体大小时，Height 和 Width 属性将自动更新。

- Left 属性和 Top 属性：Left 属性设置窗体内部最左端与屏幕最左边之间的距离，Top 属性设置窗体内部最上端与屏幕最上端之间的距离。如果设置了 StartUpPosition 属性，则这两个属性的设置有可能将不起作用。
- Picture 属性：该属性设置在窗体中显示一个图片，单击 Picture 属性右侧的按钮，将打开【加载图片】对话框，选择好对应的图片文件后单击【打开】按钮，即可为窗体设置好背景图。
- ScrollBars 属性：该属性设置窗体是否有垂直或水平滚动条，或两者都有。共有 4 种参数设置，分别为：
  - fmScrollBarsNone，值为 0，表示不显示滚动条，这是默认设置；
  - fmScrollBarsHorizontal，值为 1，表示显示水平滚动条；
  - fmScrollBarsVertical，值为 2，表示显示垂直滚动条；
  - fmScrollBarsBoth，值为 3，表示垂直和水平滚动条都显示。
- ShowModal 属性：设置窗体在显示时是模式的或者无模式的。此属性在运行时是只读的。
  - 当用户窗体是模式状态时，在使用该应用程序的任何其他部分以前都必须提供信息或者关闭用户窗体，否则将不执行后续代码，直到用户窗体被隐藏或卸载。尽管在显示一个用户窗体时，该应用程序的其他窗体无效，但其他应用程序有效。
  - 当用户窗体是无模式状态时，不关闭用户窗体也能查看其他窗体或窗口。
- StartupPosition 属性：返回或设置一个值，用来指定用户窗体第一次出现时的位置。可以用以下的值来设置 StartupPosition 属性。
  - 0-手动（Manual）：没有初始设置指定，窗体初始位置由 Left 属性和 Top 属性决定。
  - 1-所有者中心（CenterOwner）：窗体出现在使用环境的中心。
  - 2-屏幕中心（CenterScreen）：窗体出现在整个屏幕的中央。
  - 3-窗口缺省（Windows Default）：窗体以默认方式出现在屏幕的左上角。

了解用户窗体常用属性后，就可开始设置用户窗体的属性了。具体步骤如下：

（1）单击选择用户窗体，在 VBE 窗口左下方的【属性】子窗口中设置属性。

（2）在【属性】窗口中双击【（名称）】，选中右侧的默认值 UserForm1，将其修改为 frmLogin，如图 17-3 所示。

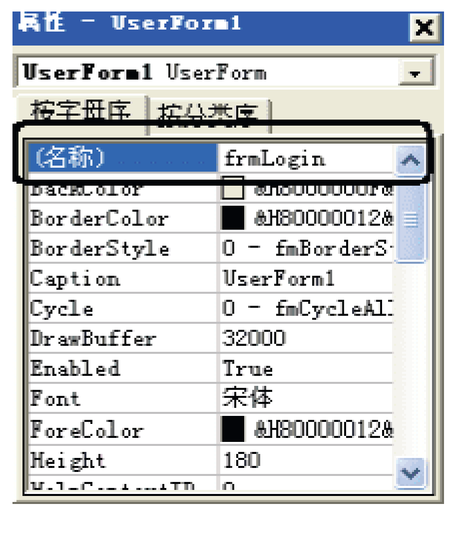


图 17-3　设置名称

（3）用类似的方法，分别设置窗体的各属性为以下值，其他值使用默认值。

- Caption 属性：登录。
- StartupPosition 属性：2-屏幕中心。

设置完属性后的用户窗体如图 17-4 所示，从外观上可以看到用户窗体的标题改为了“登录”，而 StartupPosition 属性需要在运行用户窗体时才能见到效果。

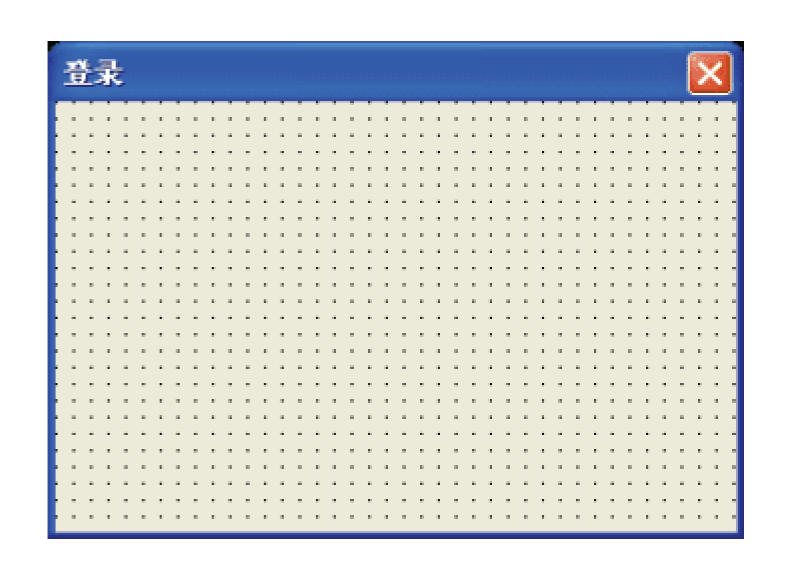


图 17-4　设置属性后的窗体

17.2　添加控件到窗体

通过上节介绍的方法，可向工程中插入用户窗体。这时的窗体还是一个空白，用户可通过窗体标题栏中的按钮来控制窗体（如移动、最小化、关闭等）。要使窗体完成与用户的交互，还需要在窗体中创建控件。

17.2.1　工具箱

当在 VBA 中向工程插入一个窗体后，将会显示出【工具箱】，在该工具箱中包含了内置的窗体控件。拖动工具箱的边框可调整其高度，如图 17-5 所示为工具箱中各控件的名称。

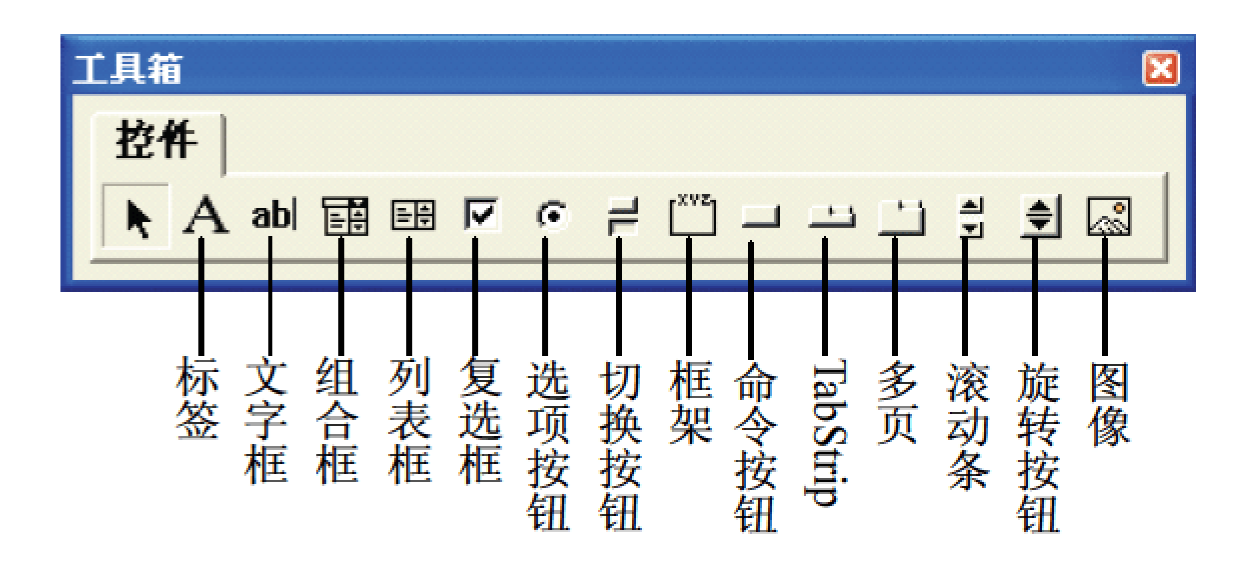


图 17-5　工具箱面板

提示：如果 VBE 窗口未显示出【工具箱】，可以单击主菜单【视图】|【工具箱】命令将其显示出来。

默认情况下，【工具箱】中显示了如图 17-5 所示的各种控件图标。在用户窗体中除了可以使用这些标准控件外，还要对其进行扩充。

在 VBA 中使用 ActiveX 控件来扩充控件工具箱，使用 ActiveX 控件的方法与使用其他标准控件的方法完全一样。在程序中加入 ActiveX 控件后，它将成为开发和运行环境的一部分，并为应用程序提供新的功能。

ActiveX 控件保留了其他标准控件的一些常用属性、方法和事件，它们的作用也相同，这样就保证了 VBA 程序的基本能力。下面以 MonthView 控件为例，演示添加控件到工具箱中的方法。

（1）右击【工具箱】的空白位置，将弹出如图 17-6 所示的快捷菜单。

（2）单击快捷菜单中的【附加控件】命令，打开如图 17-7 所示的【附加控件】对话框。

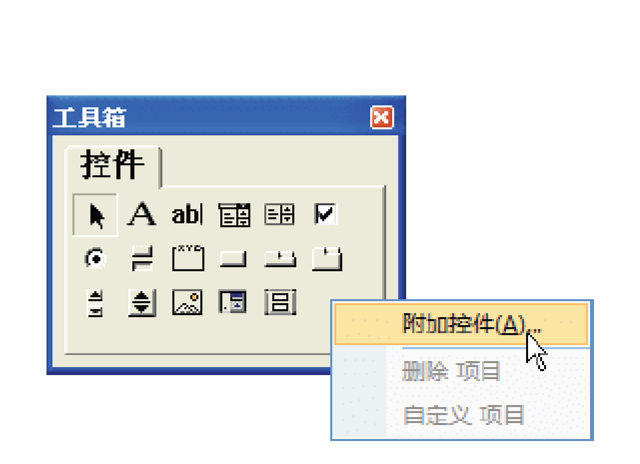


图 17-6　工具箱快捷菜单



图 17-7　附加控件

（3）在【可用控件】列表框中找到 Microsoft MonthView Control 6.0 选项，并单击选择该项数据，如图 17-8 所示。

（4）单击【确定】按钮后，工具箱中将新增加 MonthView 控件，如图 17-9 所示。



图 17-8　选择 MonthView 控件



图 17-9　添加控件后的工具箱

（5）将控件添加到工具箱后，可与使用标准控件一样，向窗体中添加日历控件。

提示：根据 Windows 操作系统安装的软件不同，不同的电脑中可能会有不同的 ActiveX 控件，常见的有日历、电子表格、图形等。

17.2.2　添加控件

向窗体中添加控件的步骤如下：

（1）在【工具箱】中单击【标签】按钮。

（2）在窗体中单击鼠标，可将【标签】控件添加到窗体中，如图 17-10 所示。

技巧：也可直接从【工具箱】中拖动控件到用户窗体来添加控件。

（3）用类似的方法，向窗体中添加两个标签、两个文本框、两个按钮后，得到如图 17-11 所示的用户窗体。

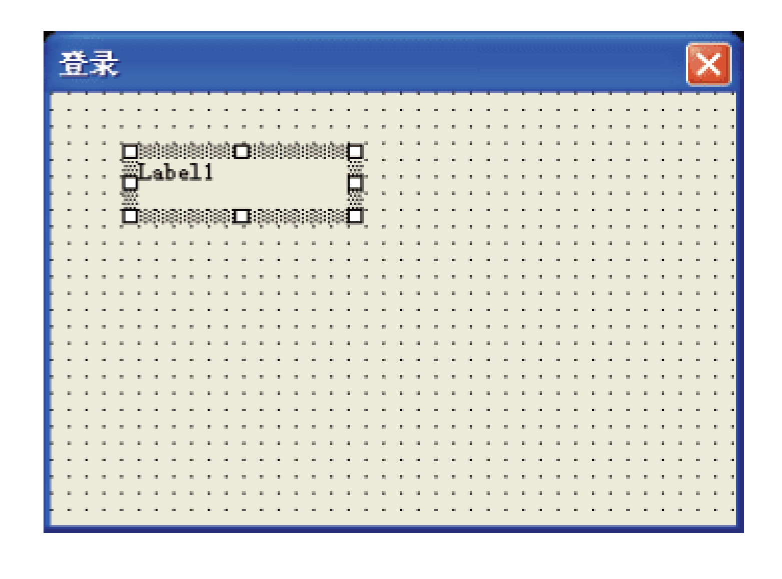


图 17-10　添加控件

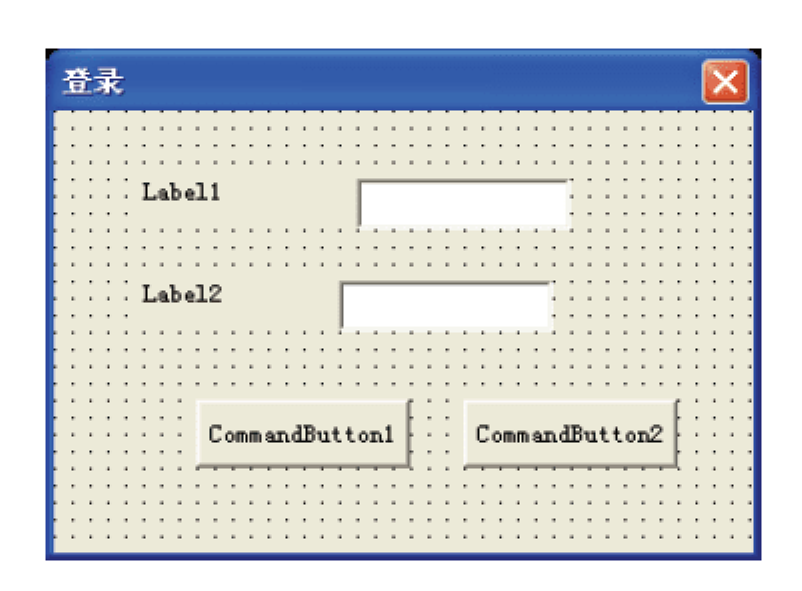


图 17-11　添加控件后的窗体

技巧：如果在窗体中需要添加多个相同的控件，可双击【工具箱】中的控件按钮，然后在窗体中多次单击鼠标即可。再次单击【工具箱】中的其他按钮，将取消对该控件按钮的按下状态。

17.3　设置控件属性

通过 17.2 节添加控件的操作，已将窗体中要使用到的控件添加到用户窗体中。这时，窗体和控件都使用默认的属性值。开发人员必须通过修改属性值，使用户窗体使用于特定的用途。例如，设置窗体名称、各相关控件显示的文字等。

提示：在实际开发过程中，一般添加一个控件就设置一个控件的属性值。

17.3.1　控件属性

对于用户窗体中的控件，具有一些相同的属性，这些常用属性如下所述。

- Name（名称）属性：该属性用来指定控件的名称，方便在程序中引用控件。
- AutoSize 属性：对于有题注的控件，AutoSize 属性规定控件是否将自动调整大小

以显示完整题注。对于没有题注的控件，该属性规定控件是否将自动调整大小以显示保存在控件中的信息。

- Caption（题注）属性：指定在控件中显示的文本。对于某些控件，则指定在标签中显示的文本。如果控件的题注太长，则此题注将被截短。如果窗体的题注太长，而题注栏又太短，此题注将带省略号显示。
- Enabled 属性：用 Enabled 属性可使控件有效或无效。无效的控件显示为浅灰色，有效控件的外观则与此不同。而且，如果控件中显示位图，则当控件变灰时位图也随之变灰。如果图像控件的 Enabled 属性为 False，那么即使该控件外观没有变灰，也不能初始化事件。
- Left 与 Top 属性：Left 属性用于设置标签与窗体左边框之间的距离。Top 属性用于设置标签与窗体上边界之间的距离。
- TabIndex 属性：指定控件在窗体 Tab 键顺序中的位置。所谓 Tab 键顺序，是在按下 Tab 键或 Shift+Tab 键后，焦点从一字段移动到下一字段的次序。
- Visible 属性：定义一个对象是可视的还是被隐藏的。将窗体上某个控件的 Visible 属性设置为 False 时，该控件将被隐藏，但仍然可通过 VBA 代码对控件的设置信息进行访问。例如，可把隐藏窗体上的控件的值作为查询的条件。

17.3.2 设置控件属性

在用户窗体中选中一个控件后，【属性】窗口将显示该控件的属性。下面开始设置【登录】用户窗体中各控件的属性。

（1）在用户窗体中单击选中名为 Label1 的标签控件。

（2）在【属性】窗口中设置 Caption 属性为“用户名：”，设置后的窗体如图 17-12 所示。

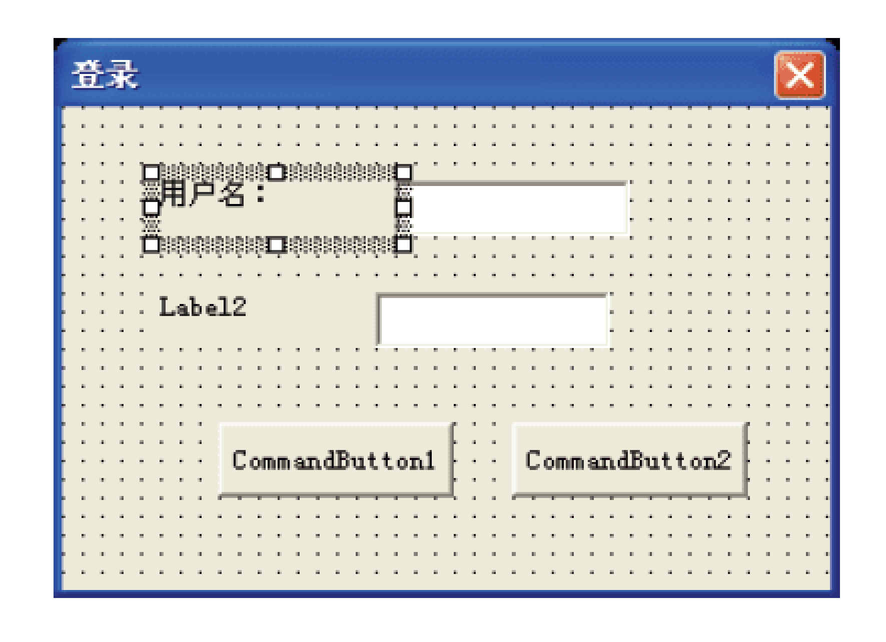


图 17-12　设置标签控件的属性

（3）用同样的方法设置标签控件 Label2 的 Caption 属性为“密码：”。

（4）选中名为 TextBox1 的文字框控件，设置该控件“（名称）”属性为 txtName。

注意：对于窗体中在以后的 VBA 代码中要访问的控件，一般应设置其“(名称)”属性为一个有意义的名称。

（5）选中名为 TextBox2 的文字框控件，设置其“（名称）”属性为 txtPwd。该文字框用来接收用户输入密码，应将用户输入的密码用一个星号（*）显示，因此需设置其 PasswordChar 属性为“*”。

（6）选中名为 CommandButton1 的命令按钮控件，设置各属性为以下值。

- “（名称）”属性：设置为 cmdLogin。
- Caption 属性：设置为登录。
- Default 属性：设置为 True。

（7）用类似的方法设置名为 CommandButton2 命令按钮的各属性：

- “（名称）”属性：设置为 cmdClose。
- Caption 属性：设置为关闭。
- Cancel 属性：设置为 True。

将各控件的属性设置完成后，得到如图 17-13 所示的用户窗体。

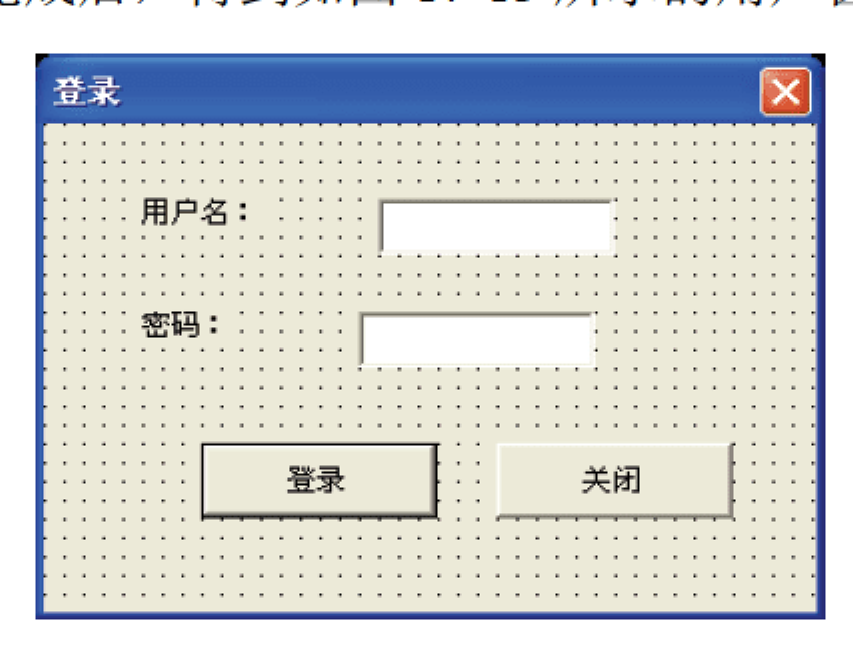


图 17-13　设置属性后的用户窗体

17.4　调整窗体中的控件

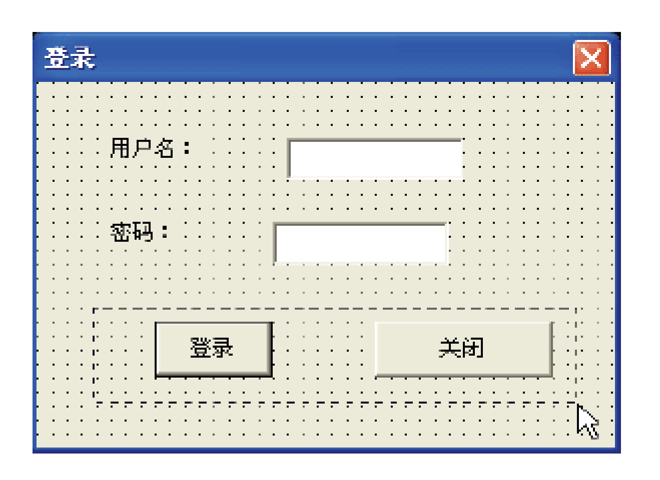
设计用户窗体时，要求尽可能美观整洁、简单实用。当用户窗体中的控件数量较多时，如何安排各控件的位置，以及设置控件的大小等操作就显得非常重要了，下面介绍快速调整控件大小及排列控件位置的方法。

17.4.1　设置控件大小

在向窗体中添加控件时，可通过拖动鼠标来控制控件的大小。对于已经添加到窗体中的控件，单击选中时其周围将出现 8 个控制点，拖动这些控制点即可设置控件的大小。

当窗体中的控件较多时，一般还需要将类似的控件设置为大小相同的外形，这时可按以下步骤操作：

（1）在窗体的空白处单击鼠标，拖拉出一个方框，被框中的控件处于选中状态，如图 17-14 左图所示。也可按住 Ctrl 键并依次单击需要设置的控件，将其全部选中，如图 17-14 右图所示。



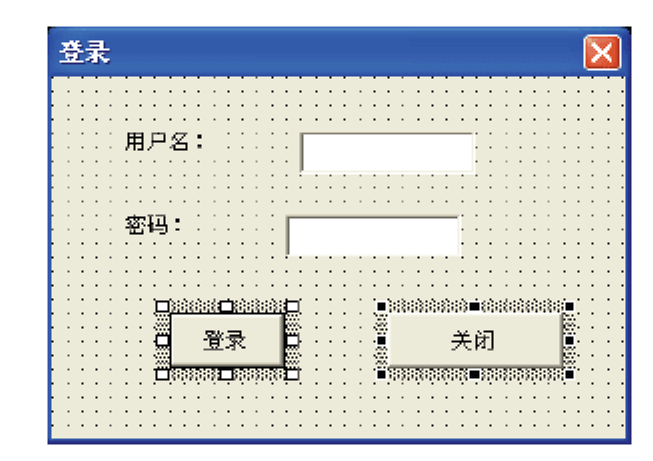


图 17-14 选中控件

注意：在图 17-14 右图中，有一个被选中的控件的控制点为白色，其他控件的控制点为黑色，表示在设置相同尺寸时，是以白色控制点控件的尺寸为准。

（2）右击任意一个被选中控件，打开如图 17-15 所示的快捷菜单，选择菜单【统一尺寸】|【两者都相同】命令，使所选控件的宽度、高度和最后一个选中控件（白色控制点）的宽度、高度相同。

技巧：单击主菜单【格式】|【统一尺寸】|【两者都相同】命令也可完成相同操作。

（3）用同样的方法可设置用户窗体的标签、文字框等控件的大小。最后得到如图 17-16 所示的效果。

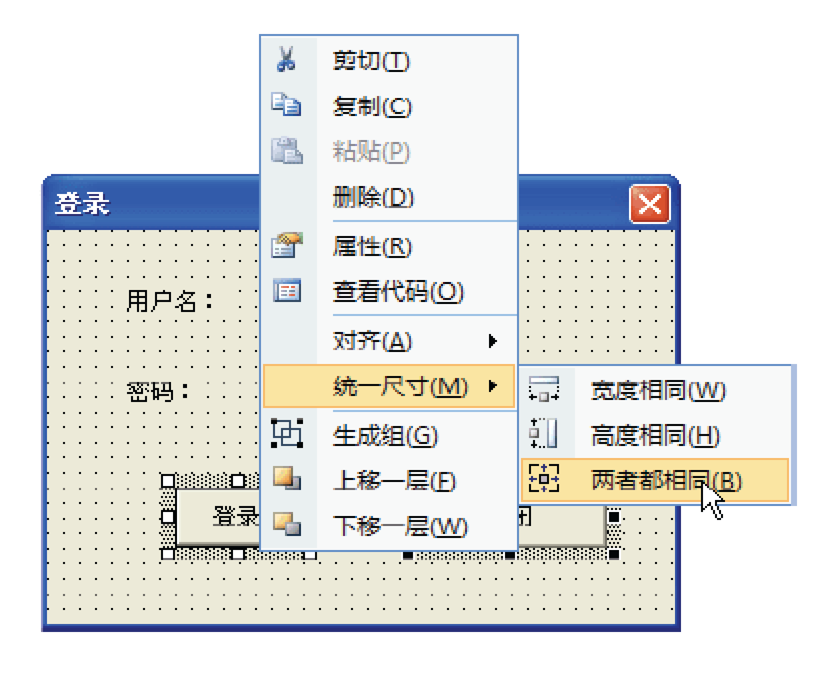


图 17-15 快捷菜单

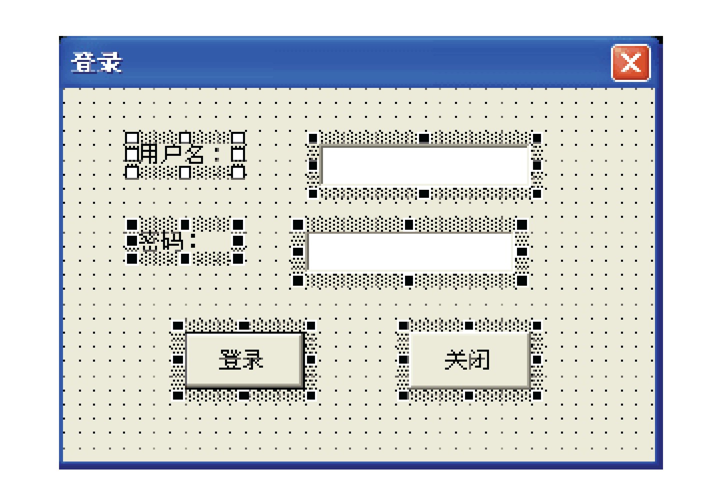


图 17-16 设置控件大小后的窗体

17.4.2 设置控件布局

在图 17-16 所示的用户窗体中，各控件的大小按需要设置好了，但可以看出各控件排列较乱，还需要对各控件的排列布局进行设置，具体步骤如下：

（1）用鼠标拖动各控件，调整控件的大概位置，如图 17-17 所示。这时，不需要精确控制各控件的位置。

（2）选中两个标签控件右键单击将弹出快捷菜单，如图 17-18 所示。

（3）在快捷菜单中单击【对齐】|【右对齐】命令。这时被选中的控件按中心线对齐（以白色框选中的按钮为基准），如图 17-19 所示。

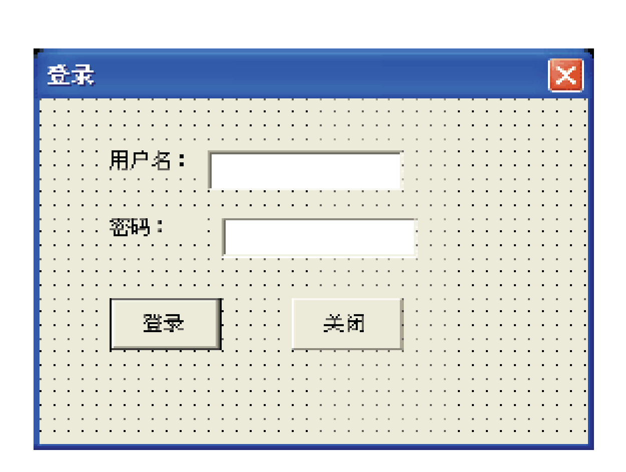


图 17-17　调整各控件的大概位置



图 17-18　快捷菜单

（4）用类似的方法设置两个文字框左对齐，两个按钮顶端（底端）对齐。这时各相关控件都已对齐，如图 17-20 所示。

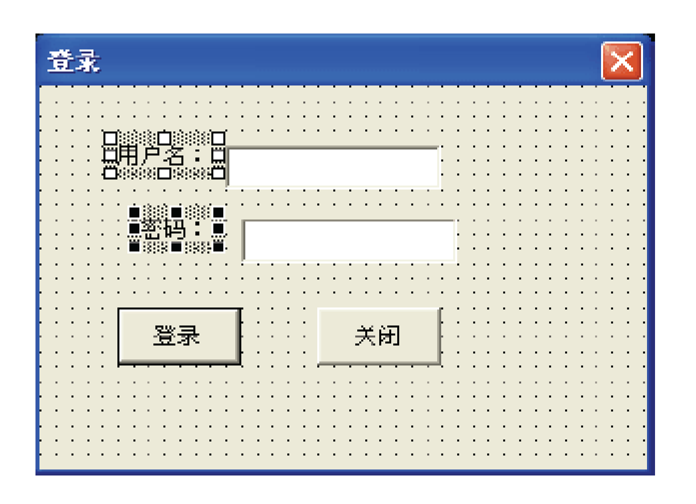


图 17-19　右对齐控件

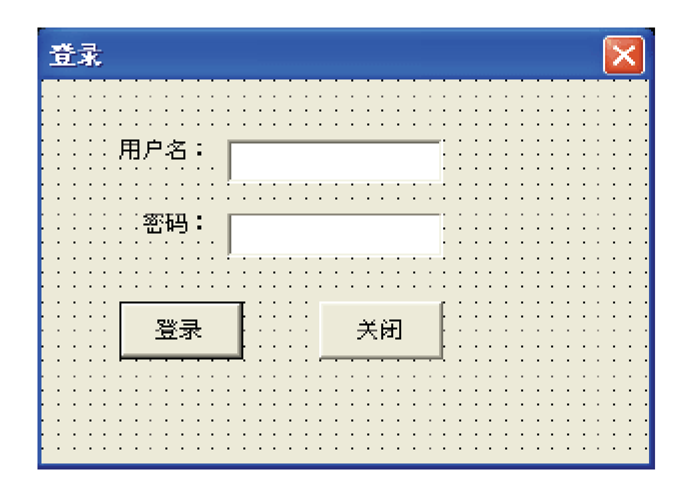


图 17-20　对齐各控件

（5）在用户窗体中选择“用户名”标签和 frmName 文字框控件，右击鼠标将弹出如图 17-21 所示的快捷菜单。

（6）在快捷菜单中单击【对齐】|【中间对齐】命令，就可以使两个控件的中心线对齐，如图 17-22 所示。



图 17-21　快捷菜单

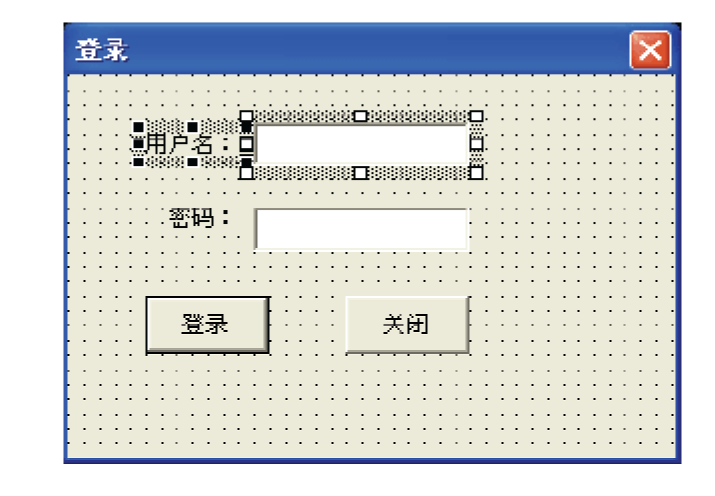


图 17-22　中间对齐

（7）用类似的方法，将各相关控件的中心线对齐。

提示： 在 VBE 中，还可单击主菜单【格式】|【垂直间距】|【相同】命令，使控件之间的垂直间距相同，单击【格式】|【水平间距】|【相同】命令，使控件之间的水平间距相同。

（8）设置好各控件的排列布局后，拖动用户窗体边框，调整窗体的大小，得到如图 17-23 所示的效果。



图 17-23　用户窗体

17.4.3　设置 Tab 键顺序

使用键盘操作控件时，按 Enter 键相当于单击具有焦点的控件。控件的焦点可使用 Tab 键来改变，Tab 键顺序确定了按 Tab 键时控件的激活顺序，还可以确定最初焦点放在哪个控件上。

要设置控件的 Tab 键顺序，可按以下步骤进行：

（1）打开用户窗体。

（2）单击主菜单【视图】|【Tab 键顺序】命令，打开如图 17-24 所示的对话框，列出了所有的控件。

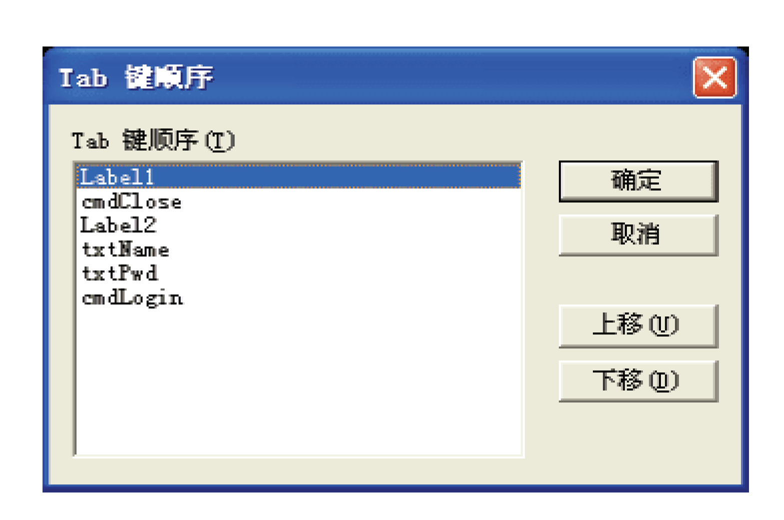


图 17-24　Tab 键顺序

（3）选中要移动顺序的某个控件。

技巧： 在图 17-24 所示的对话框中，可以按住 Ctrl 键或 Shift 键选中多个控件，再一次性移动这些控件。

（4）单击【上移】或【下移】按钮即可调整其 Tab 键顺序。

🔔提示：调整 Tab 键顺序还可以通过【属性】窗口来完成，改变 TabIndex 属性值就可以改变该控件的 Tab 键顺序。TabIndex 为 0 的控件是第一个控件。

17.5　编 写 代 码

将用户窗体及控件的属性、布局设置好以后，也就完成了用户窗体外观界面的设置。接下来更重要的工作是为用户窗体编写代码，使其能完成相应的工作。

给用户窗体编写代码可分为以下两部分：

- ❑ 给窗体中的控件编写代码；
- ❑ 给窗体事件过程编写代码。

17.5.1　编写事件代码

事件允许当用户对窗体和控件进行操作时做出相应的反应，事件程序要放置在用户窗体模块中，可以通过以下几种方式打开代码模块窗口。

- ❑ 双击窗体或控件；
- ❑ 右击窗体或控件，从快捷菜单中选择【查看代码】命令；
- ❑ 右击工程窗口中的用户窗体图标，选择【查看代码】命令。

打开代码模块窗口，如图 17-25 所示。接下来就可以对窗体或控件添加相应的事件程序代码。事件允许用户窗体和控件对用户所做的操作作出相应的反应。例如，单击用户窗体中的【关闭】按钮，将调用事件过程中的代码关闭窗体。

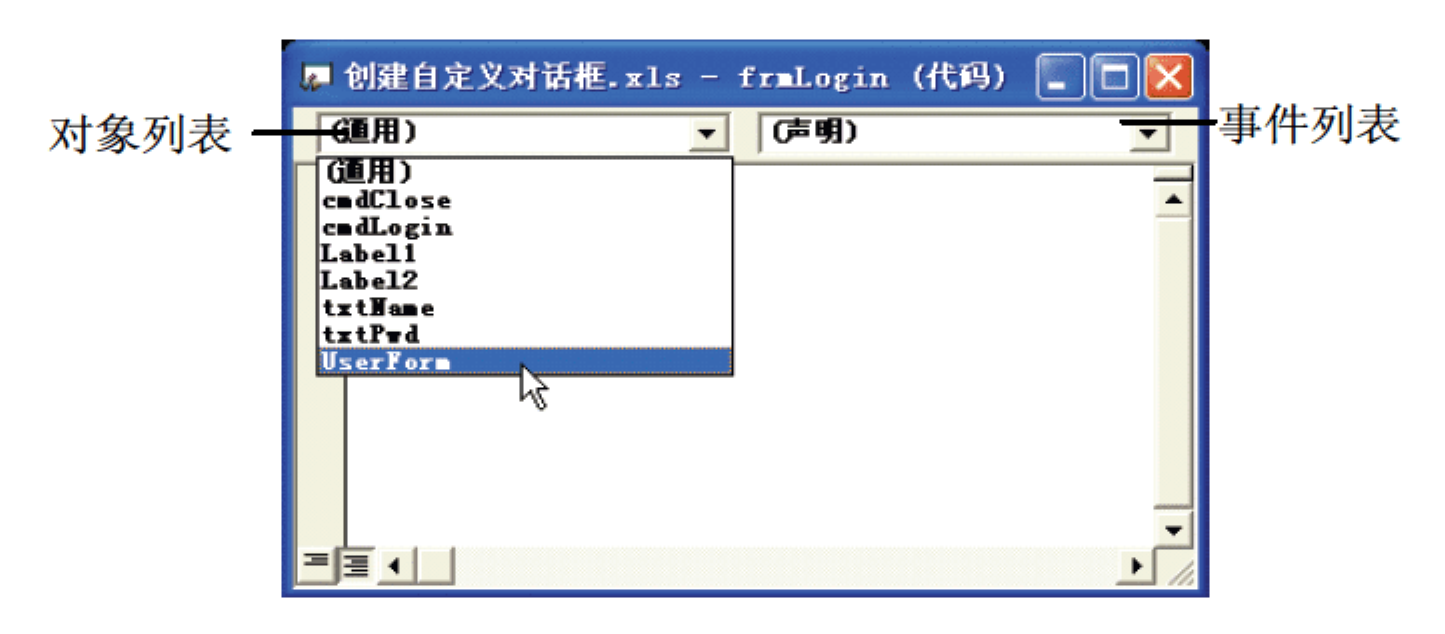


图 17-25　代码模块窗口

1．窗体事件

最常用的窗体事件包括窗体初始化（Intialize）、窗体激活（Activate）、窗体请求关闭（QueryClose）、以及单击窗体（Click）等。

- ❑ Initialize 事件：用户窗体初始化事件是窗体最重要的事件之一，初始化是发生在用户窗体中的第一件事情——只要用户窗体开始装载，就会触发初始化事件。在这里，可以初始化变量和控件。例如，可以从电子表格中更新最新的数据到文本框中，改变文本框的缺省值为今天的日期等。

- ❑ QueryClose 事件和 Terminate 事件：这两个事件是结束窗体的事件。QueryClose 请求关闭事件首先发生，并且给用户取消的机会（不会关闭窗体）；Terminate 中止事件是最终的并且不能取消。
- ❑ Activate 事件：Activate 事件激活用户窗体，如果不卸载窗体而只是隐藏窗体，再显示窗体时，初始化事件不会再运行，但是，激活（Activate）事件将会发生。当用户窗体每次获得焦点时，都会触发激活事件，在每次显示窗体时，该事件也会发生。如果有几个窗体同时可见，那么当在这些窗体之间进行切换时，激活事件也会被触发。

因此，事件的顺序是：初始化（Initialize）—激活（Activate）—处理—请求关闭（Query Close）—中止（Terminate）。

2. 控件事件

窗体中控件的常用事件包括变化（Change）、单击（Click）、输入（Enter）、退出（Exit）等。

- ❑ Change 事件：当 Value 属性的设置更改时，控件的 Change 事件发生，无论属性更改是执行代码还是用户在界面上操作的结果，此事件都发生。
- ❑ Click 事件：在导致 Click 事件发生的两种情况中，第一种情况应用于命令按钮、框架、图像、标签、滚动条和数值调节钮控件，而第二种情况用于复选框、组合框、列表框、多页、TabStrip 和切换按钮控件。当选项按钮控件的值变为 True 时，也会导致 Click 事件发生。
- ❑ Enter 事件和 Exit 事件：一个控件从同一窗体的另一个控件实际接收到焦点之前，Enter 事件发生。同一窗体中的一个控件即将把焦点转移到另一个控件之前，Exit 事件发生。

17.5.2 给控件编写代码

【登录】窗体主要用来完成用户身份的验证，编写的代码就需要验证输入的用户名和密码是否正确。编写代码的具体过程如下：

（1）双击用户窗体的空白处，打开【代码】窗口。在模块的声明部分输入以下代码，声明一个标志变量，用来记录用户是否成功登录，若其值为 True，表示已成功登录。

```
Dim bLogin As Boolean   '标志变量，记录用户是否成功登录
```

（2）在用户窗体中双击【关闭】按钮，将在【代码】窗口生成【关闭】按钮的 Click 事件过程结构，在该结构中输入卸载窗体的代码如下：

```
Private Sub cmdClose_Click()
    bLogin = False
    Unload Me
End Sub
```

（3）在用户窗体中双击【登录】按钮，将在【代码】窗口生成【登录】按钮的 Click 事件过程结构，在该结构中输入判断用户名和密码的代码如下：

```
Private Sub cmdLogin_Click()
    If Trim(txtName.Value) = "" Then    '用户名为空
        MsgBox "请输入登录用户名！", vbCritical + vbOKOnly, "警告"
        txtName.SetFocus                '用户名文字框设置焦点
        Exit Sub
    End If
    If Trim(txtPwd.Value) = "" Then     '密码为空
        MsgBox "请输入密码！", vbCritical + vbOKOnly, "警告"
        txtPwd.SetFocus                 '密码文本框设置焦点
        Exit Sub
    End If
    If txtName.Value = "admin" And txtPwd.Value = "admin888" Then
        MsgBox "欢迎进入本系统！", vbInformation + vbOKOnly, "欢迎"
        bLogin = True
        Unload Me                       '卸载窗体
    Else
        MsgBox "输入的用户名或密码有误，请重新输入！", vbCritical + vbOKOnly, "
        警告"
        txtName.SetFocus                '用户名文字框设置焦点
    End If
End Sub
```

17.5.3　编写窗体事件代码

在【登录】用户窗体中，当用户不能通过身份验证时，应退出 Excel 应用程序。在窗体各控件的代码中，都是使用 Unload Me 语句卸载用户窗体。这时，卸载窗体后将返回 Excel 操作界面。因此，需要在窗体的 QueryClose 事件中编写代码，当用户未通过身份验证而卸载窗体时，将调用 Application 对象的 Quit 方法退出 Excel。

```
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
    If Not bLogin Then          '未成功登录
        Application.Quit        '退出 Excel
    End If
End Sub
```

17.6　调用用户窗体

编写完事件代码后，需要通过运行用户窗体来测试程序的正确性。调试通过后，即可在其他模块中调用设计的用户窗体。

17.6.1　调试运行窗体

在 VBE 环境中，在窗体代码窗口中或在用户窗体设计界面中，按 F5 键或者单击工具

栏中的【运行】按钮，可以运行用户窗体程序，显示用户窗体。

也可按 F8 键逐语句运行代码，对程序进行调试。

如图 17-26 所示为测试用户窗体时的两种情况。

提示：在测试程序时，应使用多种不同的情况来验证程序是否按预设效果来运行。



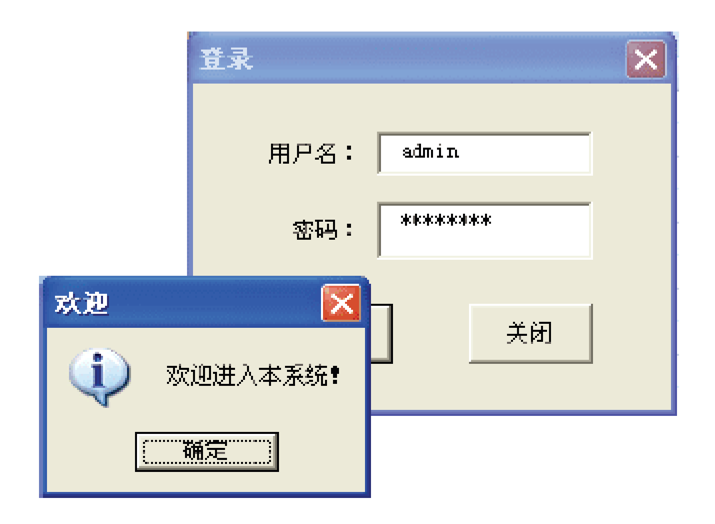


图 17-26 测试用户窗体

17.6.2 调用用户窗体基础知识

1. 窗体的生命周期

用户窗体的生命周期从其被装入内存开始，到显示、隐藏，最后到关闭为止。

窗体显示之前，必须装载到内存中。如果显示一个没有装载的窗体，该窗体将自动装载。事实上，窗体中的任何引用或者变量、控件、属性都将强制装载，并且触发初始化事件。如果想初始化窗体但不显示窗体，可以使用如下方式装载：

```
Load frmMain
```

在隐藏了窗体后，该窗体仍然被装载。如果再次显示窗体，可以使用窗体的 Show 方法。为了从内存中清除窗体，所以必须卸载（unload）窗体。当用户单击“关闭”按钮关闭窗体时，其将自动卸载。

因此，窗体装载和卸载的顺序是：装载（Load）—显示（Show）—……—隐藏（Hide）—卸载（Unload）。

2. 模式和无模式

用户窗体能在两种“模式”之间显示，即模式或者无模式。所谓模式窗体，即该窗体显示时不允许用户在 Excel 中进行其他的操作，MsgBox 对话框使用的就是模式窗体。而无模式窗体，则是在该窗体显示时还允许用户在 Excel 中进行其他操作，然后再回到该窗体中来进行操作。

当无模式窗体显示时，代码将在后台继续执行。用户可使代码暂时停止，直到窗体关闭后继续执行。

新建的用户窗体缺省设置是模式窗体。一旦窗体作为模式窗体显示后，不能将它改变为无模式窗体。如果要改变为无模式窗体，必须先隐藏该窗体，再显示它，并指定为无模式窗体。

使用 UserForm 对象的 Show 方法可以显示用户窗体，该方法的语法格式如下：

```
[object.]Show modal
```

其中参数 modal 决定用户窗体是模式的还是无模式的。其可设置为如下常数：

- vbModal，值为 1，表示用户窗体是模式的。
- vbModeless，值为 0，表示用户窗体是无模式的。

如果在运用 Show 方法时并未装载指定的对象，则 VBA 会自动装载它。

17.6.3　编写调用用户窗体的代码

可使用多种方式显示用户窗体，最常用的有以下两种方式：

- 创建一个模块，在模块中编写显示用户窗体的代码；
- 在相关的事件过程中编写显示用户窗体的代码。

1．在模块中编写代码

最常用的方法是，在模块中编写代码显示用户窗体。例如，要显示【登录】窗体，可按以下步骤编写代码。

（1）若工程还没有模块，可以单击主菜单【插入】|【模块】命令，向工程中插入一个模块。

（2）在模块中编写以下代码：

```
Sub 显示登录窗体()
    frmLogin.Show
End Sub
```

（3）关闭 VBE 窗口，返回 Excel 操作界面。在【开发工具】的【控件】组中，单击【插入】按钮，从下拉列表中选择【按钮】窗体控件。

（4）在工作表中绘制一个按钮，设置按钮调用的宏为“显示登录窗体”，并设置按钮的文本为“登录”。

（5）在工作表中单击【登录】按钮，即可显示设计的【登录】用户窗体。

2．在事件中编写代码

有的用户窗体在 Excel 应用程序的某个事件发生时自动显示。这就需要针对事件编写代码。例如，【登录】用户窗体应该是在用户打开 Excel 应用程序时就自动调用，通过该窗体对用户的身份进行认证。这时，可在 Excel 工作簿的 Open 事件中编写以下代码：

```
Private Sub Workbook_Open()
    frmLogin.Show
End Sub
```

编写以上代码后，当用户再次打开该工作簿时，将以模式窗体显示【登录】窗体。如果用户输入的用户名或密码是错误的，则将自动关闭 Excel 工作簿，不允许用户进行操作。

第 18 章　使用标准控件

在第 17 章中以创建【登录】用户窗体为例，介绍了创建用户窗体的过程。用户窗体中必须要添加合适的控件，才能完成与用户的交互操作。

当插入用户窗体后，将显示【工具箱】浮动子窗口，在该子窗口中默认的控件称为标准控件，本章将逐个介绍这些控件的属性、方法和事件。

提示：为了节省篇幅，第 17 章介绍过的控件的共有属性，本章将不再列出。

18.1　标　　签

标签控件主要用于在窗体中显示提示信息，例如各文字框、列表框等控件前面的说明文字，标签控件也可直接识别用户的单击、双击操作。

18.1.1　标签常用属性

标签控件除了具有控件的共有属性外，主要还有以下属性。

- Accelerator 属性：如果为标签指定加速键，则在 Tab 键顺序中跟在标签后面的控件接受焦点，而不是标签自身。
- BackColor 属性：设置标签的背景色。
- BorderStyle 属性：用来设置标签是否有边框，有两种值可选，即 0 代表标签无边框，1 代表标签有边框。
- Caption 属性：在用户窗体中显示的文本。
- Font 属性：用来设置标签显示的字体，既可以在设计时通过单击【属性】窗口的 Font 属性左侧的...按钮，打开显示【字体】对话框；也可以在程序中通过 VBA 代码改变标签显示的字体。
- Height 和 Width 属性：设置标签的高度和宽度，以磅为单位。
- TextAlign 属性：设置标签控件中文本的对齐方式。文本对齐方式有以下 3 种：
  - fmTextAlignLeft，值为 1，左对齐，这是默认值。
  - fmTextAlignCenter，值为 2，中对齐。
  - fmTextAlignRight，值为 3，右对齐。

18.1.2　标签事件

在实际应用中，标签控件一般只是作为控件的说明，不接受用户的输入，也不响应事件。但标签控件依然支持相应的事件，例如比较常用的 Click 事件、DbClick 事件等。

18.1.3　标签控件实例——进度条

进度条是 Windows 操作系统中常见的窗体，一般用于进行长时间操作时给用户一个提示，显示操作进行的进度。本实例创建一个进度条，用来显示正在处理工作表的进度。进度条如图 18-1 所示，在窗体中动态显示一个红色条状进度条，以显示程序运行的进度。

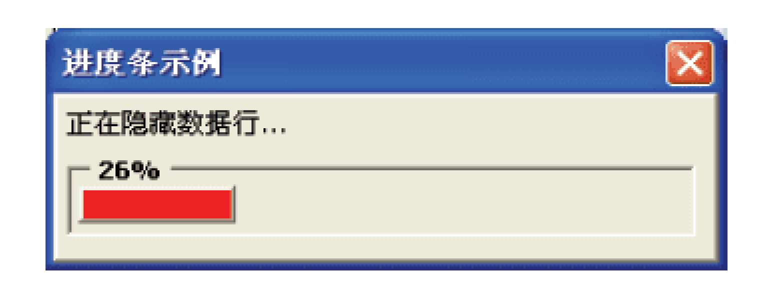


图 18-1　进度条

本例通过动态改变标签控件的宽度值，来显示程序的处理进度。主要使用标签控件的背景色和宽度属性值。完成该例的步骤如下：

（1）在 VBE 中插入一个用户窗体，向窗体中插入一个框架控件，在框架控件中再添加一个标签控件，并设置标签控件的背景色为红色。如图 18-2 所示。

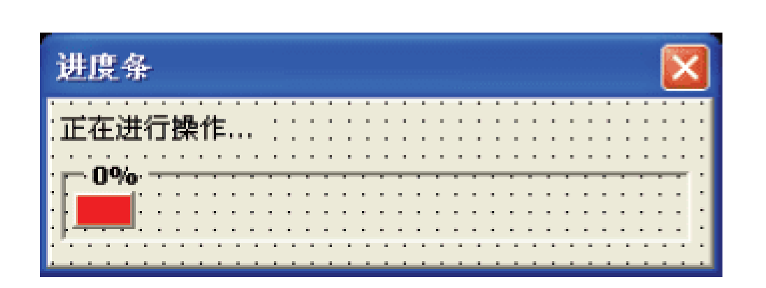


图 18-2　设置窗体

（2）双击窗体空白处，打开窗体的【代码】窗口，编写子过程 UpdateProgress，用来更新进度条的显示。

```
Sub UpdateProgress(ByVal lStart As Long, ByVal lEnd As Long, ByVal lProgress
As Long)
    p = lProgress / (lEnd - lStart)
    With Me
        .frameProgress.Caption = Format(p, "0%")
        .lblProgress.Width = p * (.frameProgress.Width - 10)
        .Repaint                        '更新显示进度条
    End With
    If p >= 1 Then Unload Me            '完成操作后，卸载窗体
End Sub
```

编写好以上代码后，在其他模块中编写代码，可以将 UpdateProgress 作为进度窗体的方法来调用。该过程提供了 3 个参数，其含义如下所述。

- lStart：处理过程的初始值；
- lEnd：处理过程的完成值；
- lProgress：正在被处理的值。

将以上 3 个参数传入 UpdateProgress 过程，该过程对这 3 个值进行运算，得出正在处理数据占总数据的比例。

（3）向工程中插入一个模块，在模块中编写完成相应功能的代码，并在每一步处理过程中，调用 frmProgress 窗体的 UpdateProgress 方法更新进度条。

```
Sub 显示进度条()
    Dim i As Long, p As Single
    With frmProgress
        .lblProgress.BackColor = RGB(255, 0, 0)
        .lblProgress.Width = 0
        .Show 0  '显示进度条
    End With

    If TypeName(ActiveSheet) <> "Worksheet" Then Exit Sub
    Application.ScreenUpdating = False
    For i = 1 To 65536
        If i Mod 2 = 0 Then
            Rows(i).Hidden = True
        End If
        frmProgress.UpdateProgress(1, 65536, i)
    Next
    Application.ScreenUpdating = True
End Sub
```

以上代码将工作表中 1～65 536 行中的偶数行隐藏起来，在 For 循环中，每执行一次循环，就调用 UpdateProgress 子过程更新进度窗口中的进度条。

该【进度条】窗体可运用到多种场合，使用时只需在代码开始处用无模式方式显示出该窗体，并在需要更新进度的地方添加以下代码即可：

```
frmProgress.UpdateProgress(lStart, lEnd, lProgress)
```

18.2 命令按钮

命令按钮是最常见的控件之一。通过单击按钮，可以触发相应的事件过程，并执行指定的操作，以实现指定的功能。例如，可以创建能够打开另一个窗体的命令按钮，在命令按钮上可以显示文本或图片，或者二者同时显示。

18.2.1 命令按钮常用属性

命令按钮除了具有控件的共有属性外，主要还有以下两个属性。

- ❑ Cancel 属性：设置命令按钮为取消按钮。如果命令按钮的 Cancel 属性被设为 True，并且是一个活动的窗体，用户就可以用鼠标单击此按钮，或按 Esc 键，或当此按钮具有焦点时再按 Enter 键，以执行此按钮的功能。
- ❑ Default 属性：设置命令按钮为默认按钮。设置 Default 属性为 True 时，按 Enter 键与单击此命令按钮的作用相同。因此，这个命令按钮被称为默认按钮。与 Cancel 的设置一样，在一个窗体中，只允许一个命令按钮的 Default 属性设置为 True。

注意：在窗体中只有一个命令按钮可作为 Cancel 按钮。若将某一命令按钮的 Cancel 属性设为 True，则窗体上所有的其他对象被自动设为 False。

18.2.2　命令按钮常用事件

命令按钮可以接受多种事件，但最常用的就是单击（Click）事件。以下是命令按钮发生单击事件的几个示例。

- ❑ 用鼠标单击命令按钮控件。如果命令按钮尚不具有焦点，则 Enter 事件发生在 Click 事件之前。
- ❑ 当命令按钮控件具有焦点时按空格键。
- ❑ 在窗体上按 Enter 键，该窗体上的一个命令按钮的 Default 属性设为 True，同时焦点没有位于其他的命令按钮上，则在该命令按钮上触发 Click 事件。
- ❑ 在一个窗体上按 Esc 键，该窗体上有一个命令按钮的 Cancel 属性设为 True，同时焦点没有位于其他的命令按钮上，则在该命令按钮上触发 Click 事件。
- ❑ 如果在设置时为命令按钮指定了加速键，则按命令按钮的加速键，也会产生 Click 事件。

18.2.3　按钮实例——控制窗体显示

本例以主窗体控制子窗体的方式，演示按钮的使用和窗体模式的相关内容。

【主窗体】对话框如图 18-3 所示，该对话框中显示了 3 个按钮，可分别进行以下操作：

- ❑ 单击【显示子窗体】按钮，将显示【子窗体】对话框，并在其中列出了显示当前子窗体的次数，如图 18-4 所示。多次单击【显示子窗体】按钮，子窗体中的显示次数将进行累加显示。
- ❑ 在【主窗体】对话框中单击【隐藏子窗体】按钮，子窗体将隐藏，再次单击【显示子窗体】按钮又将显示出子窗体，并且其显示次数增加一次。
- ❑ 在【主窗体】对话框中单击【重新设置子窗体】按钮，可重新设置子窗体的显示次数。

创建以上窗体的操作步骤如下：

（1）在 VBE 中插入一个用户窗体，设置其 Caption 属性为“主窗体”，再向窗体中添

加 3 个按钮控件，调整按钮的位置，如图 18-3 所示。设置各控件的属性如表 18-1 所示。

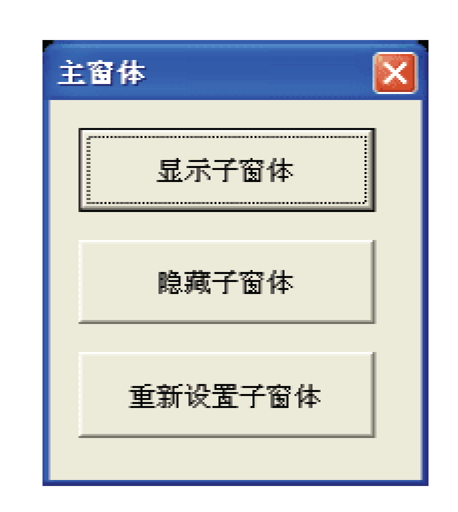




图 18-3 【主窗体】对话框

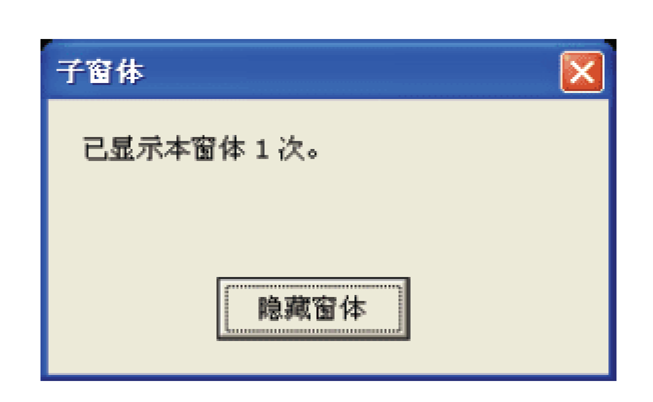


图 18-4 【子窗体】对话框

表 18-1 控件属性

用　　途	对　　象	属　　性	属　性　值
主窗体	窗体	名称	frmMain
		Caption	主窗体
显示子窗体	按钮1	名称	cmdShow
		Caption	显示子窗体
隐藏子窗体	按钮2	名称	cmdHide
		Caption	隐藏子窗体
重设子窗体	按钮3	名称	cmdReset
		Caption	重新设置子窗体

（2）再向工程中插入一个用户窗体，设置其 Caption 属性为“子窗体”，向窗体中添加一个按钮和一个标签控件，如图 18-4 所示，设置各控件的属性如表 18-2 所示。

表 18-2 控件属性

用　　途	对　　象	属　　性	属　性　值
主窗体	窗体	名称	frmSub
		Caption	子窗体
显示窗体次数	标签	名称	lbl1
隐藏窗体	按钮	名称	cmdHide
		Caption	隐藏窗体

（3）在【主窗体】对话框中，为【显示子窗体】按钮编写以下 VBA 代码：

```
Private Sub cmdShow_Click()
    Me.Hide
    If CInt(Left(Application.Version, InStr(1, Application.Version, ".", 1)
    - 1)) <= 8 Then
        'Excel97 或更早的版本不能显示无模式窗体
        frmSub.Show
        Me.Show
    Else
        '当一个模式窗体打开时,不能显示无模式窗体.因此需要使它们都为无模式窗体
```

```
        Me.Show vbModeless
        frmSub.Show vbModeless
    End If
End Sub
```

以上代码首先隐藏当前窗体，接着判断当前 Excel 的版本，若为 Excel 97 之前的版本，则按模式显示主窗体和子窗体，若是 Excel 2000 及以后版本，则用无模式方式显示主窗体和子窗体。

（4）在【主窗体】对话框中，为【隐藏子窗体】按钮编写以下 VBA 代码：

```
Private Sub cmdHide_Click()
    frmSub.Hide '隐藏子窗体
End Sub
```

（5）在【主窗体】对话框中，为【重新设置子窗体】按钮编写以下 VBA 代码：

```
Private Sub cmdReset_Click()
    Unload frmSub    '卸载子窗体
End Sub
```

以上代码卸载子窗体，从而达到重置子窗体中的模块变量的目的。

（6）在【主窗体】对话框中，为 Terminate 事件编写以下 VBA 代码，在退出该对话框之前，先卸载【子窗体】。

```
Private Sub UserForm_Terminate()
    Unload frmSub    '退出当前窗体前先关闭子窗体
End Sub
```

（7）在【子窗体】的代码窗口编写以下 VBA 代码，定义模块级变量，用来记录子窗体显示的次数。

```
Dim n As Integer    '定义模块变量
```

（8）在【子窗体】的代码窗口中为 Activate 事件过程编写以下代码，用来显示当前窗体显示的次数：

```
Private Sub UserForm_Activate()
    n = n + 1
    lbl1.Caption = "已显示本窗体 " & n & " 次。"
End Sub
```

（9）在【子窗体】的代码窗口中，给【隐藏窗体】按钮编写以下 VBA 代码：

```
Private Sub cmdHide_Click()
    Me.Hide
End Sub
```

（10）为了方便用户在 Excel 环境中打开【主窗体】对话框，可以在 VBE 中插入一个模块，编写以下子过程，用来显示【主窗体】在 Excel 工作表中插入一个按钮，并指定按钮执行以下代码。

```
Sub 控制窗体显示()
    frmMain.Show
End Sub
```

18.3 图　　像

在一些信息管理信息中，有时需要在用户窗体中显示图像。这时可使用 VBA 提供的【图像】控件。使用图像控件可剪裁、调整大小或缩放图片，但不能编辑图片的内容。

提示：标签控件也可以显示图片，但标签不支持对图片的剪裁、调整大小或缩放操作。

18.3.1 图像控件属性

图像控件支持常见的图像格式，例如*.bmp、*.cur、*.gif、*.ico、*.jpg、*.wmf 等。

除了控件共有属性外，图像控件最常用的属性还有以下几种：

- Picture 属性，通过该属性可设置图像控件显示的图片文件名。设计窗体时，可用图像控件的属性页为 Picture 属性指定图片文件名。运行窗体时，则必须使用 LoadPicture 函数为 Picture 属性指定图片文件名，其语法格式如下：

```
object.Picture = LoadPicture( pathname )
```

其中：object 为图像控件，pathname 为一个图片文件的完整路径。

要删除为控件指定的图片，可在【属性】窗口中直接删除 Picture 属性右侧的值即可，在程序代码中，将 LoadPicture 函数的 pathname 参数设置为空字符串即可。

- PictureAlignment 属性：设置图片在图像控件中的放置位置。其有以下几种方式：
  - fmPictureAlignmentTopLeft，值为 0，表示图片左上角与图像控件左上角对齐。
  - fmPictureAlignmentTopRight，值为 1，表示图片右上角与图像控件右上角对齐。
  - fmPictureAlignmentCenter，值为 2，表示图片中心与图像控件中心对齐。
  - fmPictureAlignmentBottomLeft，值为 3，表示图片左下角与图像控件左下角对齐。
  - fmPictureAlignmentBottomRight，值为 4，表示图片右下角与图像控件右下角对齐。
- PictureSizeMode 属性：设置在图像控件上显示图片的方式，可设置为以下几种方式：
  - fmPictureSizeModeClip，值为 0，表示裁掉图片中比窗体或页面大的部分（默认）。
  - fmPictureSizeModeStretch，值为 1，表示扩展图片使其填满窗体或页面。该设置值使图片在垂直和水平方向都发生变形。

- ➢ fmPictureSizeModeZoom，值为 3，表示放大图片，但图片在水平和垂直方向上都不变形。
- ❑ PictureTiling 属性：设置为 True 时，允许在窗体或页面中平铺图片，否则图片不在背景上平铺。

18.3.2　图像控件事件

与标签控件用来显示提示文字类似，图像控件一般也用来显示图片。图像控件也支持很多事件，常用的为单击（Click）事件、双击（DblClick）事件，可响应用户对图像控件的单击或双击操作。

18.3.3　图像实例——Splash 窗口

所谓 Splash 窗口，就是主界面出现之前先出现的欢迎窗口。Splash 窗体显示以后，等待一定的时间将自动显示一个消息。本例制作 Splash 窗体的步骤如下：

（1）在 VBE 中插入一个用户窗体，并向窗体中增加一个图像控件和一个标签控件，设置各控件的属性如表 18-3 所示。最后得到如图 18-5 所示的 Splash 窗体。

表 18-3　控件属性

用　　途	对　　象	属　　性	属　性　值
主窗体	窗体	名称	frmSplash
		Caption	Splash窗体
背景图像	图像	名称	Image1
		Picture	1.jpg
欢迎文字	标签	名称	Label1
		Caption	欢迎进入本系统



图 18-5　Splash 窗口

（2）双击窗体打开【代码】窗口，在窗体的 Activate 事件中编写代码，设置调用 CloseSplash 函数的时间间隔（该函数用来关闭 Splash 窗体）。具体代码如下：

```
Private Sub UserForm_Activate()
    Application.OnTime Now + TimeValue("00:00:05"), "CloseSplash"
End Sub
```

（3）在工程中插入一个模块，在模块中编写函数 CloseSplash，用来卸载 Splash 窗体，具体代码如下：

```
Private Sub CloseSplash()
    Unload frmSplash
End Sub
```

（4）在用户窗体中，为图像控件的 Click 事件编写代码，当用户单击图像控件后自动卸载当前窗体。

```
Private Sub Image1_Click()
    Unload Me
End Sub
```

（5）为了使工作簿一打开就自动打开 Splash 窗体，在【工程资源管理器】子窗体中双击 ThisWorkbook 打开代码窗口，对 Workbook 对象的 Open 事件编写以下代码，使工作簿被打开时就自动显示 frmSplash 窗体。

```
Private Sub Workbook_Open()
    frmSplash.Show
End Sub
```

18.4 文 字 框

文字框控件主要用来接收用户输入信息，也可用其显示信息供用户修改。文字框控件可接收用户输入的文本、数字、单元格引用或公式等类型的数据。

18.4.1 文字框常用属性

除了控件共有属性外，文字框控件常用的属性还有以下几种。

- ❑ EnterKeyBehavior 属性：设置在文字框中按下 Enter 键的结果。EnterKeyBehavior 属性和 MultiLine 属性是密切相关的。将 MultiLine 设为 True 时，本属性值才有具体的意义，如果本属性为 True，则按 Enter 键将创建一个新行。如果本属性设置为 False，则按 Enter 将焦点移到 Tab 键顺序的下一个对象。如果 MultiLine 为 False，则不论本属性为何值，按 Enter 键时总是把焦点移到 Tab 键顺序的下一个对象。
- ❑ MaxLength 属性：用来设置文字框中能够输入字符的最大数量。默认值为“0”，

表示在文字框中能容纳的字符数量没有限制。

- ❑ MultiLine 属性：用来设置文字框是否能够接受和显示多行文本，如果值为 True，表示文字框可容纳多行文本；如果值为 False，则表示文字框只容纳单行文本。
- ❑ PasswordChar 属性：该属性用来作为口令功能使用。例如，在【属性】窗口中将 PasswordChar 属性设置为“*”，则在文字框中无论输入什么字符，都将以“*”号显示。
- ❑ ScrollBars 属性：该属性主要用来设置文字框是否有水平或垂直的滚动条。
- ❑ Text 属性：用于显示文字框中的内容，其内容可以在设计界面时指定，也可以在程序中动态修改。在文字框控件中，Text 属性与 Value 属性的值相同。

18.4.2　文字框的方法

与前面介绍的控件不同，文字框还有 3 个方法，可用来对文字框中的内容进行操作。

- ❑ Copy 方法：将文字框中选中的文本复制到剪贴板上。
- ❑ Cut 方法：将文字框中选中的文本移至剪贴板。
- ❑ Paste 方法：把剪贴板上的内容粘贴到文字框中。

18.4.3　文字框常用事件

文字框控件主要用来接收用户的输入信息，常用的有以下几个事件。

- ❑ AfterUpdate 事件：在通过用户界面更改了文字框控件中的数据后，此事件发生。AfterUpdate 事件发生在 BeforeUpdate 事件之后。
- ❑ BeforeUpdate 事件：文字框中的数据被改变之前该事件发生。该事件代码有一个 Cancel 参数，如果在程序中将 Cancel 参数设置为 True，则焦点仍停留在该文字框控件上，并且 AfterUpdate 事件和 Exit 事件都不会发生。
- ❑ Change 事件：当文字框的内容改变时所发生的事件，常用来处理文字框内容改变后的对应策略。
- ❑ Enter、Exit 事件：一个控件从同一窗体的另一个控件实际接收到焦点之前，Enter 事件发生。同一窗体中的一个控件即将把焦点转移到另一个控件之前，Exit 事件发生。

18.4.4　文字框实例——数据输入窗体

使用文字框可接收用户输入的数据。通过文字框的相关事件，可对用户输入的数据进行判断，若输入的数据不符合要求的格式，可弹出提示信息。

本例制作如图 18-6 所示的数据输入窗体。用户在各文字框中输入数据，单击【登记】按钮，可将输入的数据填写到 Excel 工作表中。

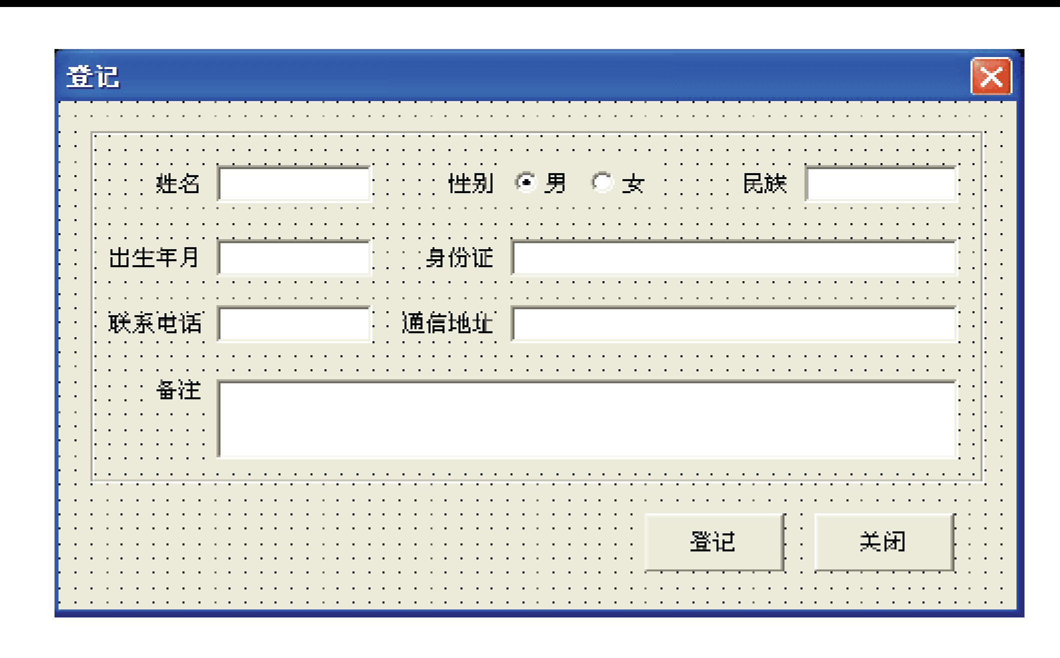


图 18-6　数据输入窗体

具体操作步骤如下：

（1）在 VBE 中插入一个用户窗体，向窗体中添加如图 18-6 所示的控件，并排列各控件的位置。设置各控件的属性如表 18-4 所示（各标签控件只用来显示文字，不列出其属性）。

表 18-4　控件属性

用　途	对　象	属　性	属　性　值
主窗体	窗体	名称	frmInput
		Caption	登记
姓名	文字框1	名称	txtName
		MaxLength	10
性别	选项按钮1	名称	optMan
		Caption	男
		Value	True
	选项按钮2	名称	optWoman
		Caption	女
民族	文字框2	名称	txtNation
出生年月	文字框3	名称	txtBirthday
		MaxLength	10
身份证	文字框4	名称	txtID
		MaxLength	18
联系电话	文字框5	名称	txtPhone
通信地址	文字框6	名称	txtAddress
备注	文字框7	名称	txtMemo
		MultiLine	True
登记	命令按钮1	名称	cmdSave
		Caption	登记
关闭	命令按钮2	名称	cmdClose
		Caption	关闭

（2）为【出生年月】文本框的 BeforeUpdate 事件编写检查代码，检查用户输入的日期是否正确，具体代码如下：

```
Private Sub txtBirthday_BeforeUpdate(ByVal Cancel As MSForms.Return
Boolean)
    If Not IsDate(txtBirthday.Value) Then
        MsgBox "请输入正确的出生年月！", , "提示"
        txtBirthday.SelStart = 0
        txtBirthday.SelLength = Len(txtBirthday.Value)
        Cancel = True
        End If
End Sub
```

（3）为【身份证】文本框的 BeforeUpdate 编写以下代码，检查身份证号码的正确性：

```
Private Sub txtID_BeforeUpdate(ByVal Cancel As MSForms.ReturnBoolean)
    Dim strId As String '暂存身份证号
    strId = txtID.Value
    If Len(strId) <> 15 And Len(strId) <> 18 Then
        MsgBox "身份证号码位数错误，只能为 15 位或 18 位！", , "提示"
        txtID.SelStart = 0
        txtID.SelLength = Len(strId)
        Cancel = True
    End If
End Sub
```

（4）为【登记】按钮编写代码，将用户窗体中输入的内容添加到工作表中，具体代码如下：

```
Private Sub cmdSave_Click()
    If txtName.Value = "" Then
        MsgBox "请输入员工姓名！", , "提示"
        txtName.SetFocus
        Exit Sub
    End If
    add
End Sub
```

以上代码首先检查，如果“姓名”为空，则退出保存数据的过程。最后调用自定义函数 add 完成添加过程。

（5）自定义函数 add 的作用，是将用户在窗体中输入的内容按一定的对应关系添加到 Excel 工作表中，具体代码如下：

```
Sub add()
    Dim intRow As Integer
    With Sheets("登记表")
        intRow = .Range("A1").CurrentRegion.Rows.Count  '获取已有数据行数
        intRow = intRow + 1                              '将用户窗体上的数据
                                                          填充到新行上

        .Cells(intRow, 1) = txtName.Value                '姓名
        If optMan.Value Then
            .Cells(intRow, 2) = "男"
        Else
            .Cells(intRow, 2) = "女"
```

```
        End If
        .Cells(intRow, 3) = txtNation.Value          '民族
        .Cells(intRow, 4) = txtBirthday.Value        '出生年月
        .Cells(intRow, 5).NumberFormatLocal = "@"    '设置身份证列为文本格式
        .Cells(intRow, 5) = txtID.Value              '身份证号码
        .Cells(intRow, 6).NumberFormatLocal = "@"    '设置联系电话为文本格式
        .Cells(intRow, 6) = txtPhone.Value           '联系电话
        .Cells(intRow, 7) = txtAddress.Value         '通信地址
        .Cells(intRow, 8) = txtMemo.Value            '备注
    End With
End Sub
```

（6）最后，为【关闭】按钮编写以下代码，卸载当前窗体：

```
Private Sub cmdClose_Click()
    Unload Me    '卸载窗体
End Sub
```

18.5 复 选 框

复选框用来表示一个特定的状态是选定还是清除。在应用程序中为用户提供从两种方案中选其一的操作。由于复选框彼此独立工作，所以用户可同时选择任意多个复选框。

18.5.1 复选框属性

除了具有控件的共有属性外，复选框控件的其他常用属性分别如下所述。

- Picture 属性：可为复选框控件指定一个显示的图片。
- Value 属性：用来设置返回控件的状态。当选中复选框控件时，该属性为 True，而属性值为 False 时，则表示没有选择该复选框。
- TripleState 属性：决定用户能否在用户界面为复选框指定 Null 状态。其值为 True 时，表示复选框控件可在 3 个状态中选择。而其值为 False 时，则复选框控件只支持 True 和 False 状态。

18.5.2 复选框事件

复选框最主要的事件是 Click 事件，选中控件时，其 Value 值变为 True。在运行程序时，如果使用代码改变复选框的 Value 属性时，也会发生 Click 事件。

18.5.3 复选框实例——设置 Excel 选项

在 Excel 操作环境中，打开【Excel 选项】对话框，可看到该对话框中使用了很多的复选框控件，如图 18-7 所示。

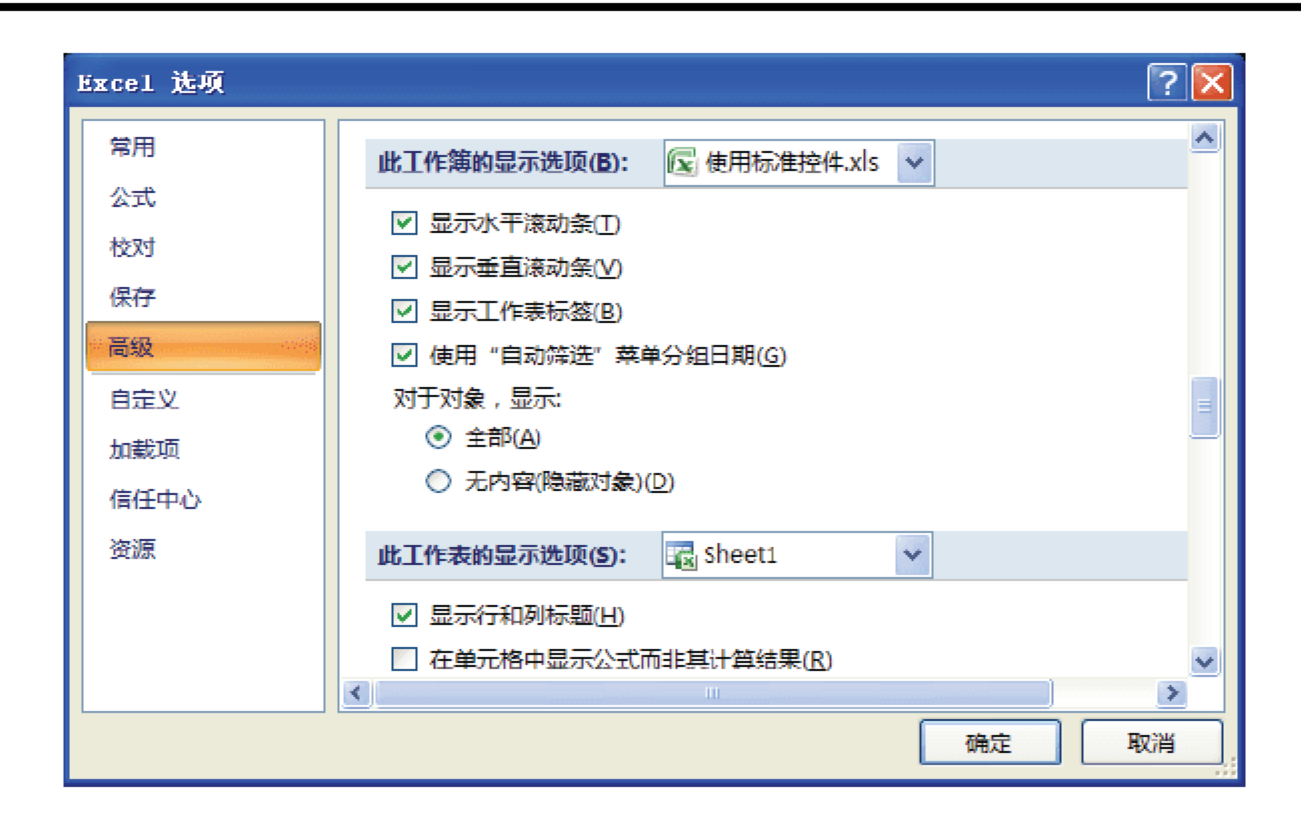


图 18-7 【Excel 选项】中的复选框

本例创建用户窗体用来设置 Excel 的部分选项，完成的用户窗体如图 18-8 所示。

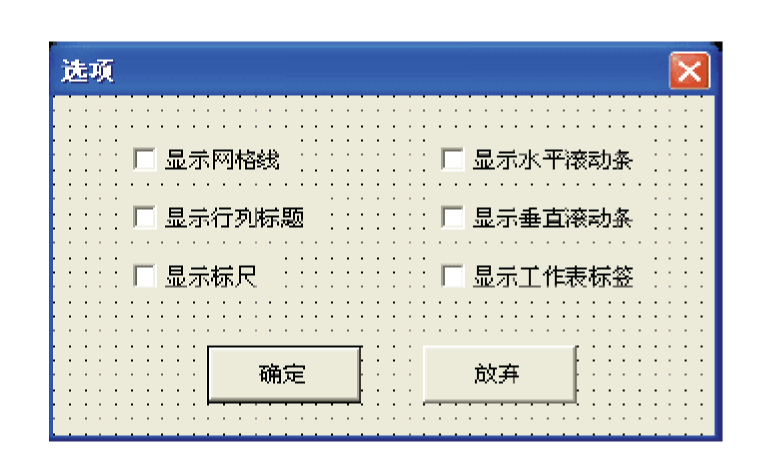


图 18-8 【选项】用户窗体

制作如图 18-8 所示的【选项】用户窗体的步骤如下：

（1）在 VBE 中单击主菜单【插入】|【用户窗体】命令，向工程中增加一个用户窗体。

（2）向用户窗体中增加 6 个复选框控件，2 个按钮控件，以调整控件的布局如图 18-8 所示，并设置各控件的属性如表 18-5 所示。

表 18-5 控件属性

用　途	对　象	属　性	属 性 值
主窗体	窗体	名称	frmOption
		Caption	选项
网格线	复选框1	名称	chkGridlines
		Caption	显示网格线
行列标题	复选框2	名称	chkHeadings
		Caption	显示行列标题
标尺	复选框3	名称	chkRuler
		Caption	显示标尺
水平滚动条	复选框4	名称	chkHScrollBar
		Caption	显示水平滚动条
垂直滚动条	复选框5	名称	chkVScrollBar
		Caption	显示垂直滚动条

续表

用　　途	对　　象	属　　性	属　性　值
工作表标签	复选框6	名称	chkTabs
		Caption	显示工作表标签
确定	命令按钮1	名称	cmdOK
		Caption	确定
		Default	True
放弃	命令按钮2	名称	cmdCancel
		Caption	放弃
		Cancel	True

（3）双击用户窗体，进入【代码】窗口，为窗体的 Initialize（初始化）事件编写代码，获取 Excel 中相关选项的值，并使用该值初始化用户窗体中的各复选框控件。

```
Private Sub UserForm_Initialize()
   With ActiveWorkbook.Windows(1)
      chkGridlines.Value = .DisplayGridlines
      chkHeadings.Value = .DisplayHeadings
      chkRuler.Value = .DisplayRuler
      chkHScrollBar.Value = .DisplayHorizontalScrollBar
      chkVScrollBar.Value = .DisplayVerticalScrollBar
      chkTabs.Value = .DisplayWorkbookTabs
   End With
End Sub
```

（4）当用户设置不同复选框的值后，单击【确定】按钮，即可用复选框中的值来修改 Excel 中相关选项的值。

```
Private Sub cmdOK_Click()
   With ActiveWorkbook.Windows(1)
      .DisplayGridlines = chkGridlines.Value
      .DisplayHeadings = chkHeadings.Value
      .DisplayRuler = chkRuler.Value
      .DisplayHorizontalScrollBar = chkHScrollBar.Value
      .DisplayVerticalScrollBar = chkVScrollBar.Value
      .DisplayWorkbookTabs = chkTabs.Value
   End With
   Unload Me
End Sub
```

本例使用的各选项设置值都为 True 或 False，而复选框的 Value 属性也可返回 True 或 False。因此，本例可直接将复选框的 Value 属性值给相关选项的属性值即可。

18.6 选项按钮

选项按钮控件与其他程序设计语言中的单选按钮相同。选项按钮是分组的，在一组中

只有一个按钮处于选中或按下状态。在应用程序中要从几个选项中选一个时，就可以用到该选项按钮。

18.6.1　选项按钮常用属性

除了具有控件共有属性外，选项按钮常用的属性还包括以下几个。

- GroupName 属性：创建一个互斥的选项按钮控件组。要创建一个互斥的数值调节按钮控件组，可将那些按钮放到窗体上的一个框架中，也可以使用 GroupName 属性。
- Picture 属性：可为选项按钮指定一个显示的图片。
- Value 属性：用来设置返回控件的状态。当选项按钮值为 True 时，表示已选择了该按钮；值为 False 时，表示没有选择该选项按钮。
- TripleState 属性：决定用户能否在用户界面为选项按钮指定 Null 状态。其值为 True 时，选项按钮可在 3 个状态中选择。而其值为 False 时，按钮只支持 True 和 False 状态。

18.6.2　选项按钮常用事件

选项按钮的常用事件是 Click 事件，在运行程序时，如果设置选项按钮的 Value 属性值为 True，则会发生 Click 事件。用户单击选项按钮也将发生 Click 事件。

18.6.3　选项按钮实例——设置窗体字号和颜色

选项按钮也是最常见的控件之一。本例制作如图 18-9 所示的用户窗体，在该窗体中，用户可通过单击【字号】和【颜色】选项按钮，分别设置文字框中文字的大小和颜色。

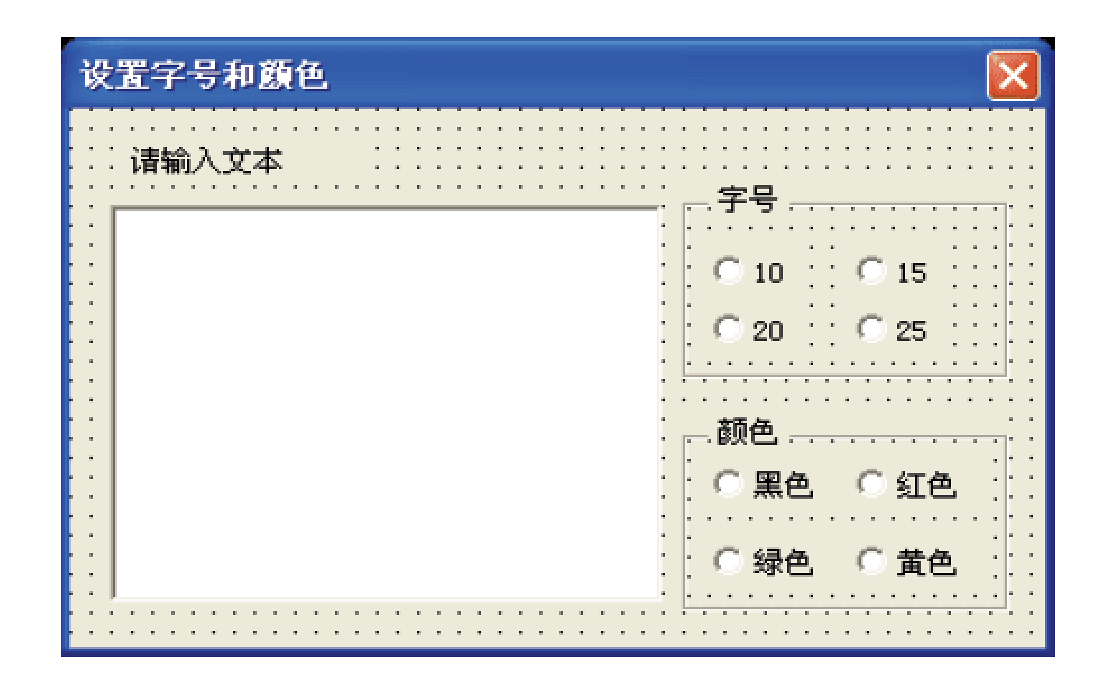


图 18-9 【设置字号和颜色】对话框

制作上图所示用户窗体的步骤如下：

（1）在 VBE 中，向工程中插入一个用户窗体。

（2）向用户窗体中添加 1 个标签控件、1 个文字框控件、2 个框架控件和 8 个选项按钮控件。调整各控件的布局如图 18-9 所示，并设置各控件的属性值如表 18-6 所示。

表 18-6　控件属性

用　　途	对　　象	属　　性	属　性　值
主窗体	窗体	名称	frmFontsize
		Caption	设置字号和颜色
提示信息	标签	Caption	请输入文本
接收用户输入文字	文字框	名称	txtFont
		MultiLine	True
字号分组	框架1	Caption	字号
字号大小	选项按钮1	名称	opt10
		Caption	10
	选项按钮2	名称	opt15
		Caption	15
	选项按钮3	名称	opt20
		Caption	20
	选项按钮4	名称	opt25
		Caption	25
颜色分组	框架2	Caption	颜色
颜色	选项按钮5	名称	optBlack
		Caption	黑色
	选项按钮6	名称	optRed
		Caption	红色
	选项按钮7	名称	optGreen
		Caption	绿色
	选项按钮8	名称	optYellow
		Caption	黄色

注意：本例使用框架来为两组不同的选项按钮分组。这时，需首先向窗体中添加框架控件，接着选中框架控件后，再向框架中添加选项按钮。

（3）设置好各控件的属性后，就可以开始编写代码了。本例直接在选项按钮的 Click 事件中编写代码，当用户单击该选项按钮时即可执行该事件过程中的代码。首先在字号为 10 的按钮的 Click 事件过程中编写以下代码：

```
Private Sub opt10_Click()
    txtFont.Font.Size = 10
End Sub
```

其他字号选项按钮的代码与此类似，就不再列出了。

（4）为颜色框架中的【黑色】选项按钮 Click 事件编写代码如下：

```
Private Sub optBlack_Click()
    txtFont.ForeColor = vbBlack
```

```
End Sub
```

其他颜色选项按钮的代码与此类似。

18.7 列　表　框

列表框主要用来显示项目列表，当项目总数超过控件高度时，列表框控件会自动添加垂直方向的滚动条。用户可从列表框中选择一项或多项数据。

18.7.1　列表框常用属性

列表框与文字框最主要的区别在于列表框能够显示多行文本，并且每行文本是一个可以独立处理的项。除控件共有属性之外，列表框常用属性还包括以下几个。

- ❑ BoundColumn 属性：列表框可以显示多列数据，当选择了多列列表框的一行时，BoundColumn 属性标识出将该行的哪一条目作为控件的值存储。
- ❑ ColumnCount 属性：设置列表框显示的列数。
- ❑ ColumnHeads 属性：该属性设置是否显示列表框的列标题行。
- ❑ ColumnWidths 属性：指定多列列表框中的各列的宽度。ColumnWidths 属性中的任意或全部设置值均可为空。仅输入列表分隔符而没有预置值便可建立一个空设置。
- ❑ List 属性：用来获取或设置列表框中的列表项内容。List 属性是字符串数组，每个数组元素都是列表框中的一个列表项，通过 List（下标值）的形式表示。如列表框中的第 1 项，用 List（0）表示，列表框中的第 2 项，用 List（1）表示，列表框中的第 10 项，用 List（9）表示，以此类推。
- ❑ ListCount 属性：ListCount 属性常与 List 属性一起使用，表示列表框中有多少个列表项。
- ❑ ListIndex 属性：返回列表框中选中选项的序号。
- ❑ MultiSelect 属性：是否允许列表框进行多选。可设置为以下常数。
  - ➢ FmMultiSelectSingle：值为 0，表示只可选择一个条目，这是默认值。
  - ➢ FmMultiSelectMulti：值为 1，表示按空格键或单击鼠标以选定列表中一个条目或取消选定。
  - ➢ FmMultiSelectExtended：值为 2，按 Shift 键并单击鼠标，或按 Shift 键的同时按一个方向键，将所选条目由前一项扩展到当前项。按 Ctrl 键的同时单击鼠标可选定或取消选定。
- ❑ Selected 属性：Selected 属性表示列表框中各个列表项是否被选中，它是一个数组，其数组元素的个数与列表框的项数相同。

提示：当 MultiSelect 属性设置为 Extended 或 Simple ，必须用列表框的 Selected 属性确定选定的条目。而且，控件的 Value 属性总是 Null。

18.7.2　列表框的方法

列表框提供相应的方法，用来操作列表框中的列表框。

- ❑ AddItem 方法

用 AddItem 可以为列表框增加项目，其代码规则如下：

```
[对象名].AddItem[项字符串] [，索引值]
```

其中，“项字符串”是用双引号（""）定界的，表示将要在列表中显示的项名称。“索引值”是从 0 开始的顺序号，标明新增的项在列表框中的位置，如果没有“索引值”，则表示将新增的项添加到列表末尾。

- ❑ Clear 方法

用 Clear 可以清除列表框中的所有内容，其代码规则如下：

```
[列表框名称].Clear
```

- ❑ RemoveItem 方法

RemoveItem 方法能够从列表框中删除某一项，其一般格式如下：

```
[对象名]. RemoveItem[索引值]
```

其中，索引值是必须的，表示欲删除哪一个项目。

对于任意一个列表框，要删除已经选中的项目，代码如下：

```
[列表框名称].RemoveItem[列表框名称].ListIndex。
```

18.7.3　列表框实例——列表框间移动数据

在用户窗体中使用列表框可显示大量数据，特别适合显示工作表中的数据。本例从 Excel 工作表中读入数据，添加到列表框，再通过用户窗体中的相关按钮，分别操作列表框中的数据。本例的工作界面如图 18-10 所示。

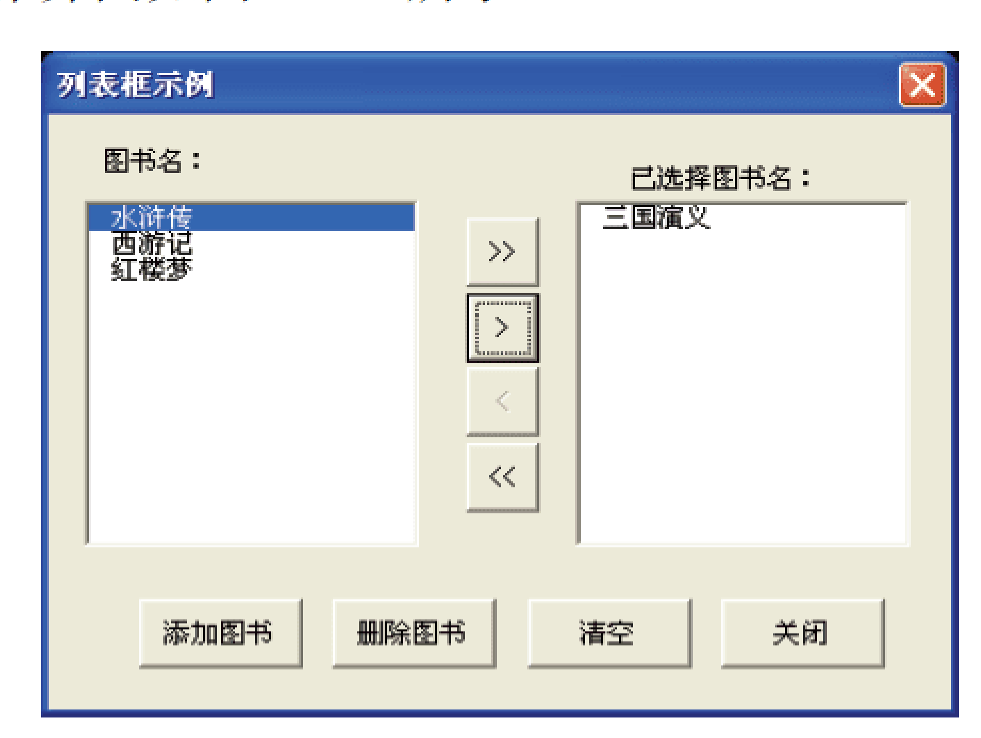


图 18-10 【列表框示例】对话框

如图 18-10 所示，用户窗体从工作表“图书名”中获取相关数据，添加到左侧列表框中。通过两个列表框中部的 4 个按钮可在两个列表框之间移动数据，单击下方的【添加图书】按钮可向左侧列表框中添加数据，单击【删除图书】按钮可将左侧列表框中选择的数据项删除，单击【清空】按钮可清除两个列表框中的数据项。

要完成以上用户窗体，可按以下步骤操作：

（1）在 VBE 中插入一个用户窗体，向窗体中添加 2 个列表框控件和 8 个命令按钮控件，调整各控件布局，如图 18-10 所示，并设置各控件的属性，如表 18-7 所示。

表 18-7　控件属性

用　　途	对　　象	属　　性	属　性　值
主窗体	窗体	名称	frmList
		Caption	列表框示例
左侧列表框	列表框1	名称	lstLeft
右侧列表框	列表框2	名称	lstRight
右移列表框所有数据	命令按钮1	名称	cmdToRightAll
		Caption	>>
右移列表框选中数据	命令按钮2	Caption	cmdToRight
		Caption	>
左移列表框选中数据	命令按钮3	名称	cmdToLeft
		Caption	<
左移列表框所有数据	命令按钮4	名称	cmdToLeftAll
		Caption	<<
添加数据	命令按钮5	名称	cmdAdd
		Caption	添加图书
删左侧列表框选中数据	命令按钮6	名称	cmdDel
		Caption	删除图书
清空两个列表框	命令按钮7	名称	cmdClear
		Caption	清空
关闭	命令按钮8	名称	cmdClose
		Caption	关闭
		Cancel	True

（2）在窗体的 Initialize 事件中编写以下代码，用来从工作表中添加数据到列表框：

```
Private Sub UserForm_Initialize()
    Dim r1 As Range, i As Long

    With ActiveWorkbook.Worksheets("图书名")
        i = .Range("A1").End(xlDown).Row             '已有数据行数
        Set r1 = .Range(Cells(2, 1), Cells(i, 1))    '获取已有数据引用
    End With
```

```
    For i = 1 To r1.Rows.Count                          '添加到列表框
        lstLeft.AddItem r1(i)
    Next

    If lstLeft.ListCount >= 1 Then                      '设置各按钮的可用性
        cmdToRight.Enabled = True
        cmdToRightAll.Enabled = True
    Else
        cmdToRight.Enabled = False
        cmdToRightAll.Enabled = False
    End If
    cmdToLeft.Enabled = False
    cmdToLeftAll.Enabled = False
End Sub
```

以上代码首先从 Excel 工作表“图书名”的 A 列获取数据，添加到窗体左侧的列表框中，最后根据列表框中列表框的数据项数量，设置各按钮的可用状态。

（3）在左侧列表框和右侧列表框的单击事件中编写代码，用来设置两个列表框之间按钮的可用状态，具体代码如下：

```
Private Sub lstLeft_Click()
    If lstLeft.ListIndex >= 0 Then
        cmdToRight.Enabled = True
        cmdToRightAll.Enabled = True
    Else
        cmdToRight.Enabled = False
        cmdToRightAll.Enabled = False
    End If
End Sub

Private Sub lstRight_Click()
    If lstRight.ListIndex >= 0 Then
        cmdToLeft.Enabled = True
        cmdToLeftAll.Enabled = True
    Else
        cmdToLeft.Enabled = False
        cmdToLeftAll.Enabled = False
    End If
End Sub
```

（4）单击 cmdToRightAll 按钮时，将左侧列表框中的数据全部移至右侧列表框中，具体代码如下：

```
Private Sub cmdToRightAll_Click()
    Do While lstLeft.ListCount > 0                '循环处理所有数据
        lstRight.AddItem lstLeft.List(0)          '添加列表框中第 1 项
        lstLeft.RemoveItem 0                      '删除第 1 项
```

```
    Loop
                                        '以下代码设置按钮状态
    If lstRight.ListCount > 0 Then cmdToLeftAll.Enabled = True
    cmdToRight.Enabled = False
    cmdToRightAll.Enabled = False
End Sub
```

以上代码通过循环操作，每次移动左侧列表框中的第 1 个列表框，使用 List(0)可访问第 1 个列表框。

（5）单击 cmdToRight 按钮时，将左侧列表框中选中的数据项移到右侧列表框中，具体代码如下：

```
Private Sub cmdToRight_Click()
    If lstLeft.ListIndex >= 0 Then
        lstRight.AddItem lstLeft.Text               '在右边列表中增加新元素
        lstLeft.RemoveItem lstLeft.ListIndex        '删除左边列表中的所选元素
    End If
    If lstRight.ListCount > 0 Then cmdToLeftAll.Enabled = True
    If lstLeft.ListCount > 0 Then
        cmdToRight.Enabled = True
        cmdToRightAll.Enabled = True
    Else
        cmdToRight.Enabled = False
        cmdToRightAll.Enabled = False
    End If
End Sub
```

（6）单击名为 cmdToLeft 的按钮，将右侧列表框中选中的数据项移到左侧列表框中，具体代码如下：

```
Private Sub cmdToLeft_Click()
    If lstRight.ListIndex >= 0 Then
        lstLeft.AddItem lstRight.Text               '在右边列表中移动数据到左侧
        lstRight.RemoveItem lstRight.ListIndex      '删除右边列表中的所选元素
    End If
    If lstLeft.ListCount > 0 Then cmdToRightAll.Enabled = True
    If lstRight.ListCount > 0 Then
        cmdToLeft.Enabled = True
        cmdToLeftAll.Enabled = True
    Else
        cmdToLeft.Enabled = False
        cmdToLeftAll.Enabled = False
    End If
End Sub
```

（7）单击名为 cmdToLeftAll 的按钮时，将右侧列表框中的数据全部移到左侧列表框中，具体代码如下：

```
Private Sub cmdToLeftAll_Click()
    Do While lstRight.ListCount > 0
        lstLeft.AddItem lstRight.List(0)
        lstRight.RemoveItem 0
    Loop
    If lstLeft.ListCount > 0 Then cmdToRightAll.Enabled = True
    cmdToLeft.Enabled = False
    cmdToLeftAll.Enabled = False
End Sub
```

（8）单击名为 cmdAdd 的按钮时，将弹出一个输入对话框，让用户输入新的数据，并将该数据添加到左侧列表框中，具体代码如下：

```
Private Sub cmdAdd_Click()
    Dim strItem As String    '
    strItem = InputBox("在列表中输入新图书名：")    '向列表中输入新项目

    If Trim(strItem) <> "" Then
        lstLeft.AddItem strItem
    End If
End Sub
```

（9）单击名为 cmdDel 的按钮时，将删除左侧列表框中选中的数据项。

```
Private Sub cmdDel_Click()
    If lstLeft.ListIndex >= 0 Then
        lstLeft.RemoveItem lstLeft.ListIndex
    End If
End Sub
```

（10）单击名为 cmdClear 的按钮时，将清除两个列表框中所有的数据项。

```
Private Sub cmdClear_Click()
    lstLeft.Clear
    lstRight.Clear
End Sub
```

18.8 复 合 框

复合框控件将文字框和列表框的特性结合在一起，既可在控件中的文字框部分输入信息，也可在控件的列表框部分选择某项信息。

18.8.1 复合框常用属性

列表框控件的大部分属性同样适合复合框，此外复合框还有自己的一些属性。

❑ ListRows 属性：指定复合框的下拉列表中显示的最大行数。如果列表中的条目数

超出 ListRows 属性的值，则将在复合框的列表框部分的右边出现一个滚动条。

- Style 属性：VBA 中的复合框有两种不同的形式，不同的 Style 属性值确定了复合框的类型和显示方式。
  - FmStyleDropDownCombo：值为 0，称为“下拉式复合框”，它由可输入的编辑区和一个下拉式列表框组成。用户可以从键盘直接向文本编辑区输入内容，也可单击三角形按钮，从下拉列表中选择。
  - FmStyleDropDownList：值为 2，称为“下拉式列表框”，其外形与“下拉式复合框”相似，但用户只能从列表框中选择而不能直接向文本区输入。
- Text 属性：该属性用来返回选择的文本或直接在编辑区输入的文本，可在界面设置时直接输入。

18.8.2　复合框常用方法

跟列表框一样，复合框也适用 AddItem、Clear、RemoveItem 方法。

18.8.3　复合框常用事件

根据复合框的类型，它们所响应的事件有所不同。

例如，当复合框的 Style 属性为 0 或 2 时，能够接收 Click 与 Dropdown 事件；当 Style 属性为 0 时，文字框还可以接收 Change 事件。

18.8.4　复合框实例——微机配置单

下面以如图 18-11 所示的用户窗体为例，来演示复合框的使用。

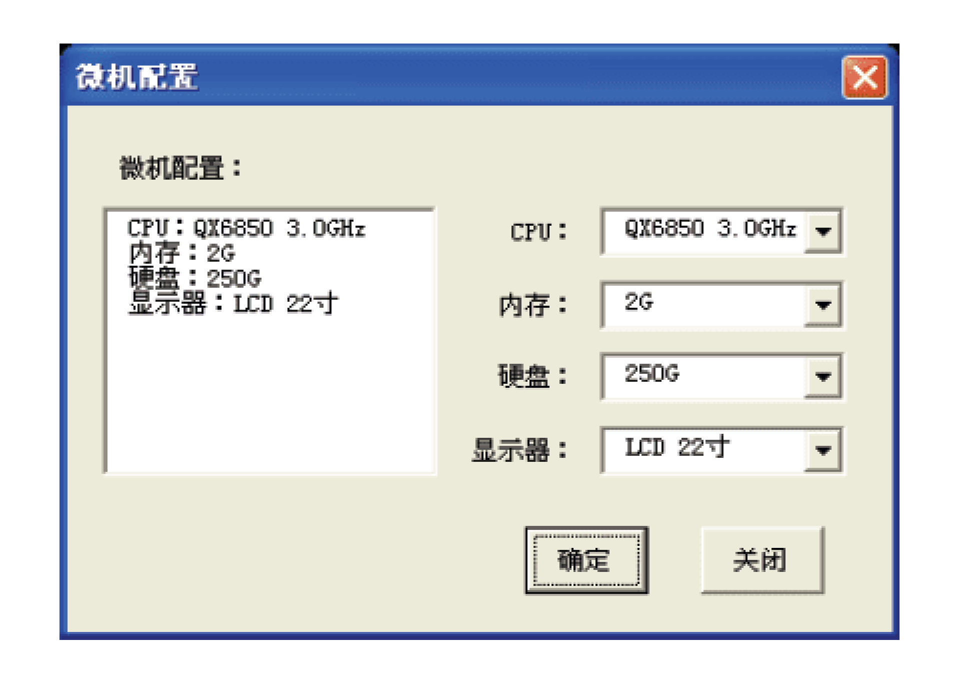


图 18-11　复合框示例

制作图 18-11 所示用户窗体的具体步骤如下：

（1）在 VBE 中插入一个用户窗体，在窗体中放置 5 个标签控件、1 个文字框控件、4 个复合框控件、2 个命令按钮控件。调整各控件的布局如图 18-11 所示，设置各控件的属性如表 18-8 所示。

表 18-8　控件属性

用　　途	对　　象	属　　性	属　性　值
主窗体	窗体	名称	frmComputer
		Caption	微机配置
显示配置	文字框1	名称	txtComputer
		MultiLine	True
CPU列表	复合框1	名称	cmbCPU
内存列表	复合框2	名称	cmbMem
硬盘列表	复合框3	名称	cmbHDD
显示器列表	复合框4	名称	cmbDisp
确定	命令按钮1	名称	cmdAdd
		Caption	确定
		Default	True
关闭	命令按钮2	名称	cmdClose
		Caption	关闭
		Cancel	True

（2）双击窗体的空白处打开【代码】窗口，在窗体的初始化事件中编写以下代码，向各复合框中添加初始数据。

```
Private Sub UserForm_Initialize()
    With cmbCPU
        .AddItem "QX6850 3.0GHz"
        .AddItem "QX6800 2.93GHz"
        .AddItem "QX6700 2.66GHz"
        .AddItem "X7800 2.6GHz"
        .AddItem "X6800 2.93GHz"
        .AddItem "E6850 3.0GHz"
        .AddItem "E6750 2.66GHz"
        .AddItem "E6700 2.66GHz"
        .Text = .List(0)
    End With
    With cmbMem
        .AddItem "512M"
        .AddItem "1G"
        .AddItem "2G"
        .AddItem "3G"
        .AddItem "4G"
        .Text = .List(0)
    End With

    With cmbHDD
        .AddItem "80G"
        .AddItem "120G"
        .AddItem "160G"
        .AddItem "250G"
```

```
        .Text = .List(0)
    End With

    With cmbDisp
        .AddItem "CRT 19 寸"
        .AddItem "CRT 21 寸"
        .AddItem "LCD 17 寸"
        .AddItem "LCD 19 寸"
        .AddItem "LCD 22 寸"
        .Text = .List(0)
    End With
End Sub
```

（3）当用户在各复合框中选择好需要的数据后，单击【确定】按钮，可在左侧的文字框中显示所选择的微机配置列表。该按钮的代码如下：

```
Private Sub cmdAdd_Click()
    Dim str1 As String
    str1 = "CPU: " & cmbCPU.Value & Chr(13)
    str1 = str1 & "内存: " & cmbMem.Value & Chr(13)
    str1 = str1 & "硬盘: " & cmbHDD.Value & Chr(13)
    str1 = str1 & "显示器: " & cmbDisp.Value
    txtComputer.Value = str1
End Sub
```

18.9　滚　动　条

滚动条是可放置在窗体中的独立控件。它看上去与某些控件（例如列表框或复合框的下拉部分）中的滚动条类似。然而，与这些例子中的滚动条不同，独立的滚动条不是任何其他控件的组成部分。

与 VB 不同，VBA 工具箱中只提供了一种滚动条，要创建横向或纵向滚动条，通过将已放置在窗体中的滚动条进行水平或垂直拖动，就可以创建横向或纵向的滚动条。

18.9.1　滚动条常用属性

除控件共有的属性外，滚动条控件常用的属性还有下述的几个。

- LargeChange 属性：当用户单击滚动条和滚动箭头之间的区域时，返回滚动条控件的 Value 属性值的改变量。
- SmallChange 属性：当用户单击滚动箭头时，返回滚动条控件的 Value 属性值的改变量。
- Max 属性：当滚动框处于底部或最右位置时，返回一个滚动条位置的 Value 属性最大设置值。
- Min 属性：当滚动框处于顶部或最左位置时，返回一个滚动条位置的 Value 属性最

小设置值。

- ❑ Value 属性：滚动条的当前位置，其返回值始终介于 Max 和 Min 属性值之间，包括这两个值。

🔔注意：Min 和 Max 属性的范围为–32768～32767，默认设置 Max 为 32767，Min 为 0。如果 Max 被设为比 Min 小的值，那么最大值将被分别设为水平或垂直滚动条的最左或最上位置处。

18.9.2　滚动条常用事件

捕获滚动条的事件，可使滚动条修改的数值进行即时响应。滚动条控件的常用事件分别如下所述。

- ❑ Change 事件：移动滚动条的滚动框部分，在进行滚动或通过代码改变 Value 属性的设置时发生。
- ❑ Scroll 事件：当 ScrollBar 控件上的滚动框被重新定位，或按水平方向或垂直方向滚动时，将发生 Scroll 事件。

🔔注意：Change 是滚动完成后触发的事件，Scroll 是在滚动过程中触发的事件。

18.9.3　滚动条实例——显示比例

在 Excel 中打开【显示比例】对话框如图 18-12 所示，在该对话框中可设置工作表的显示比例。但该对话框操作不太方便，本例制作一个自定义的【显示比例】对话框，如图 18-13 所示。在该对话框中，用户可通过单击滚动条来调整工作表的显示比例，也可在下方的文本框中输入显示比例的数值。在该对话框右侧，还可以单击水平和垂直滚动条来滚动工作表，以显示不同的区域。





图 18-12　系统的【显示比例】对话框

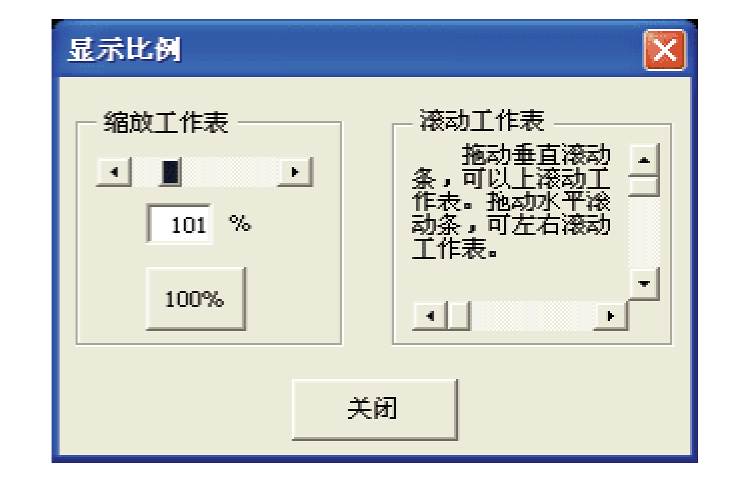


图 18-13　自定义的【显示比例】对话框

制作图 18-13 所示对话框的具体步骤如下：

（1）在 VBE 中插入一个用户窗体，向窗体中添加 2 个框架控件、3 个滚动条控件、一个文字框控件和 1 个按钮控件，调整各控件布局如图 18-13 所示，设置各控件的属性如表 18-9 所示。

表 18-9　控件属性

用　　途	对　　象	属　　性	属　性　值
主窗体	窗体	名称	frmZoom
		Caption	显示比例
缩放工作表	框架1	名称	Frame1
		Caption	缩放工作表
	滚动条1	名称	scbZoom
		Max	400
		Min	10
		Value	100
	文字框	名称	txtZoom
	命令按钮1	名称	cmd100
滚动工作表	滚动条1	名称	scbH
	滚动条2	名称	scbV
关闭	命令按钮2	名称	cmdClose
		Caption	关闭
		Cancel	True

（2）在窗体的初始化事件中编写以下代码，设置各滚动条的初始值：

```
Private Sub UserForm_Initialize()
   txtZoom.Value = ActiveWindow.Zoom                  '文字框显示当前比例

    With scbZoom                                      '缩放滚动条的属性
        .Min = 10
        .Max = 400
        .SmallChange = 1
        .LargeChange = 10
        .Value = ActiveWindow.Zoom
    End With

    With scbH                                         '水平滚动工作表参数
        .Min = 1
        .Max = ActiveSheet.Cells.Columns.Count         '最大列数
        .Value = ActiveWindow.ScrollColumn             '当前列
        .LargeChange = 10
        .SmallChange = 1
    End With

    With scbV                                         '垂直滚动工作表参数
        .Min = 1
        .Max = ActiveSheet.Cells.Rows.Count            '最大行数
        .Value = ActiveWindow.ScrollRow                '当前行
        .LargeChange = 10
        .SmallChange = 1
    End With
End Sub
```

（3）为【缩放工作表】滚动条的 Change 事件编写代码，对当前窗口进行缩放。具体代码如下：

```
Private Sub scbZoom_Change()
    With ActiveWindow
        .Zoom = scbZoom.Value          '用滚动条的值设置当前窗口的缩放
        txtZoom = .Zoom                '设置文字框的值
        .ScrollColumn = scbH.Value     '最左边的列号
        .ScrollRow = scbV.Value        '最顶端的行号
    End With
End Sub
```

（4）为【100%】按钮的单击事件编写 VBA 代码，设置缩放滚动条的值为 100，此时将产生缩放滚动条的 Change 事件，对当前窗口进行缩放。具体代码如下：

```
Private Sub cmd100_Click()
     scbZoom.Value = 100
End Sub
```

（5）为输入缩放比例的文本框的 AfterUpdate 事件编写以下 VBA 代码，将文本框中输入的值赋值为缩放滚动条：

```
Private Sub txtZoom_AfterUpdate()
    scbZoom.Value = txtZoom.Value
End Sub
```

（6）为窗体右侧【滚动工作表】框架中的水平和垂直滚动条的 Change 事件编写代码，设置当前窗体的 ScrollColumn 或 ScrollRow 属性值为滚动条的值。具体代码如下：

```
Private Sub scbH_Change()
    ActiveWindow.ScrollColumn = scbH.Value
End Sub

Private Sub scbV_Change()
    ActiveWindow.ScrollRow = scbV.Value
End Sub
```

18.10 旋转按钮

旋转按钮控件主要用来输入一定范围内的整数值，一般将旋转按钮与文字框结合起来使用。单击旋转按钮只会更改旋转按钮的值，还需要编写代码修改与之结合的文字框的值。

18.10.1 旋转按钮常用属性

旋转按钮控件最常用的是 Max 和 Min 属性，这两个属性规定控件的 Value 属性可接收的最大值和最小值。单击旋转按钮控件的上下箭头按钮可改变控件的 Value 属性。

18.10.2　旋转按钮常用事件

当旋转按钮控件 Value 属性改变时，将触发 Change 事件，一般在该事件中编写代码，来获取控件的 Value 值，并应用到与之关联的文字框控件中。

18.10.3　旋转按钮实例——修改日期和时间

本例制作如图 18-14 所示的用户窗体，使用旋转按钮来调整日期和时间值。

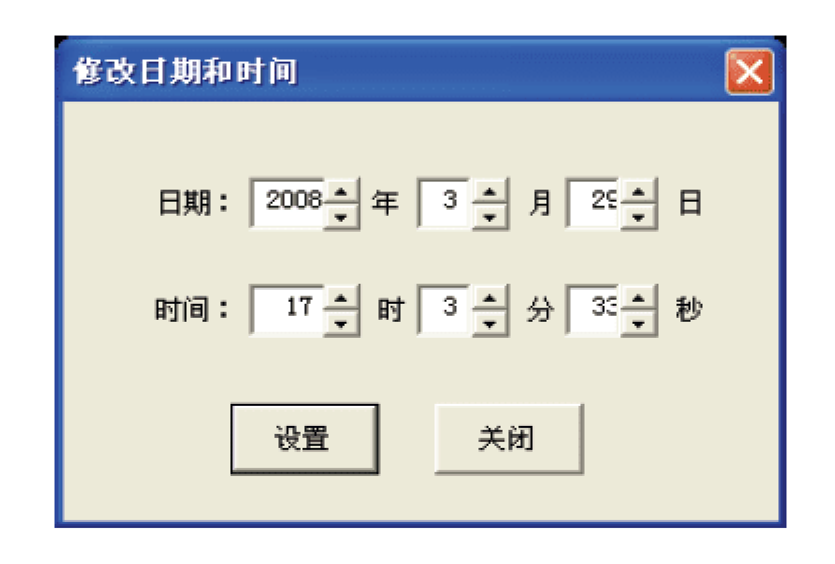


图 18-14 【修改日期和时间】对话框

制作以上用户窗体的具体操作步骤如下：

（1）在 VBE 中插入一个用户窗体，在窗体中添加 8 个标签、6 个文字框、6 个旋转按钮和 2 个命令按钮。调整各控件的布局如图 18-14 所示，设置各控件的属性如表 18-10 所示。

表 18-10　控件属性

用　　途	对　　象	属　　性	属　性　值
主窗体	窗体	名称	frmDate
		Caption	修改日期和时间
显示年	文字框1	名称	txtYear
		MaxLength	4
显示月	文字框2	名称	txtMonth
		MaxLength	2
显示日	文字框3	名称	txtDay
		MaxLength	2
显示时	文字框4	名称	txtHour
		MaxLength	2
显示分	文字框5	名称	txtMinute
		MaxLength	2
显示秒	文字框6	名称	txtSecond
		MaxLength	2

续表

用　　途	对　　象	属　　性	属　性　值
调整年	旋转按钮1	名称	spbYear
		Min	1900
		Max	2500
调整月	旋转按钮2	名称	spbMonth
		Min	1
		Max	12
调整日	旋转按钮3	名称	spbDay
		Min	1
		Max	31
调整时	旋转按钮4	名称	spbHour
		Min	0
		Max	23
调整分、秒	旋转按钮5 旋转按钮6	名称	spbMinute、spbSecond
		Min	0
		Max	59

提示： 在调整日期的旋转按钮中，初始设置其 Max 属性为 31，在程序中根据选择的年和月决定每月的天数，使其作为 Max 属性的值。

（2）在窗体的初始化事件中编写以下代码，获取系统当前日期和时间，分别显示在对应的文字框中，并根据当前的值初始化各旋转按钮的值。

```
Private Sub UserForm_Initialize()
    txtYear.Value = Year(Date)            '获取当前日期的年月日
    txtMonth.Value = Month(Date)
    txtDay.Value = Day(Date)
    txtHour.Value = Hour(Time)            '获取当前时间的时分秒
    txtMinute.Value = Minute(Time)
    txtSecond.Value = Second(Time)

    spbYear.Value = txtYear.Value         '为旋转按钮赋值
    spbMonth.Value = txtMonth.Value
    spbDay.Value = txtDay.Value

    spbHour.Value = txtHour.Value
    spbMinute.Value = txtMinute.Value
    spbSecond.Value = txtSecond.Value
End Sub
```

（3）给名为 spbYear 的旋转按钮控件的 Change 事件编写以下代码，用旋转按钮的值更新对应文字框中的值。

```
Private Sub spbYear_Change()
```

```
    txtYear.Value = spbYear.Value
End Sub
```

提示：窗体中其他旋转按钮的 Change 事件代码与此类似，这里不再一一列出。

（4）在日期设置中，根据月大或月小的不同，每月天数也不相同。对于闰年，2 月的天数也不相同。这时，可以编写代码计算每月的天数。在本例中，使用以下代码来生成每月的天数。其计算方法时，取下月 1 日减去 1 天，即可得到上月的最后一天，再由 Day 函数得到该天的日期即可。取得当月的天数后，通过调整日期的旋转按钮的 Max 属性限制其最大值。

```
Private Sub txtMonth_Change()
    Dim dTemp As Date
    dTemp = DateSerial(txtYear.Value, txtMonth.Value + 1, 1)
    spbDay.Max = Day(dTemp - 1)
End Sub
```

（5）用户在窗体中设置好日期和时间值后，单击【设置】按钮，即可用输入的参数设置系统的日期和时间。该按钮 Click 事件的代码如下：

```
Private Sub cmdSet_Click()
    Dim dTemp, dTime
    dTemp = DateSerial(txtYear.Value, txtMonth.Value, txtDay.Value)
                        '生成日期
    dTime  =  TimeSerial(txtHour.Value,  txtMinute.Value,  txtSecond.Value)
                        '生成时间

    Date = dTemp        '设置日期
    dTime = dTime       '设置时间
End Sub
```

18.11　多　　页

多页控件可以在窗体中显示一系列不同的页面，在处理可以划分为不同类别的大量信息时很有用。利用多页控件能够将相关信息组织在一起显示出来，同时又能够随时访问整条记录。

18.11.1　多页控件常用属性

多页控件对象为 MultiPage，在该控件中还包括有子对象——Page 对象。在窗体中单击多页控件对象时，将选中当前的 Page 对象，可设置 Page 对象的属性，Page 对象类似于一个单独的窗体，每个 Page 对象都包括一套自己的控件，并且不需要依赖于集合中其他 Page 对象的信息。

1．MultiPage对象属性

若要设置 MultiPage 对象的属性，可单击控件的外边框（移动多页控件时也需要拖动此边框），这时【属性】窗口中显示的将是 MultiPage 对象。MultiPage 对象的常用属性如下：

- ❑ MultiRow 属性：设置控件是否有多行标签，当 MultiRow 属性为 True 时将标签分行显示，为 False 时，标签将在一行中显示，右侧将出现左右滚动的箭头。
- ❑ Value 属性：标识当前激活页。0 表示是第一页。通过访问该属性，在程序中可获得当前激活页，从而进行适当的操作。也可通过给 Value 属性设置新的值来激活对应的页。

2．Page对象属性

Pages 集合包含多页控件中的所有页面。多页中的每个页面都是一个窗体，每个窗体都可包含自己的控件，并且可以有唯一的布局。一般情况下，多页中的页面都有标签，以便让用户选择单个页面。

默认情况下，每个多页控件包含两个页面，称作 Page1 和 Page2。每个页面都是一个 Page 对象，合在一起就表示多页的 Pages 集合。

Page 对象的常用属性如下所述。

- ❑ ScrollBars 属性：设置页面是否有垂直或水平滚动条，或两者都有。可为以下值。
  - ➢ FmScrollBarsNone：不显示滚动条（默认）。
  - ➢ FmScrollBarsHorizontal：显示水平滚动条。
  - ➢ FmScrollBarsVertical：显示垂直滚动条。
  - ➢ FmScrollBarsBoth：垂直和水平滚动条都显示。
- ❑ TransitionEffect 属性：设置从一页转换成另一页时，所用的可视效果。
- ❑ TransitionPeriod 属性：以毫秒为单位定义一个页面过渡效果的历时长度。

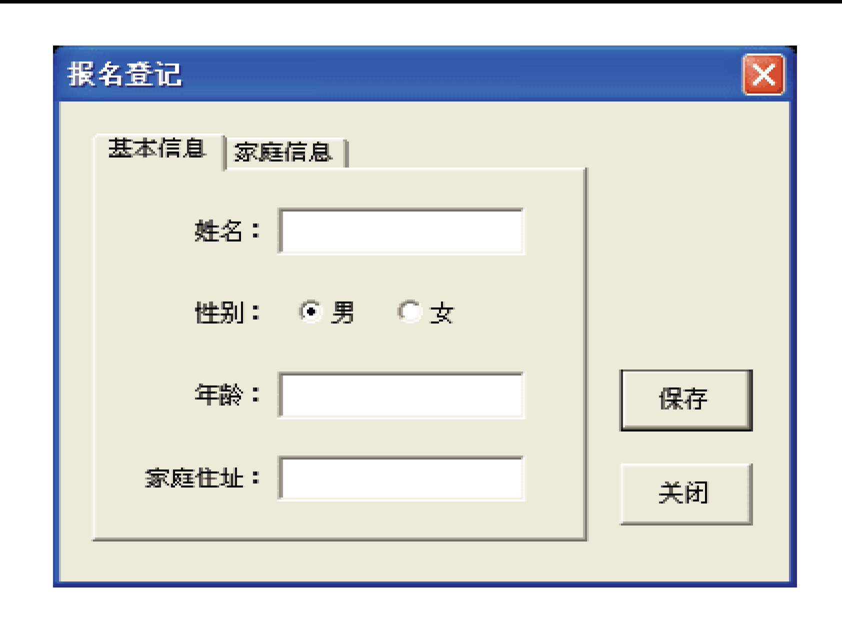
18.11.2 多页控件常用事件

多页控件支持单击（Click）等事件，常用的是 Change 事件，当选择不同的页（改变 Value 属性值）时将触发 Change 事件，此时 Value 值将是新活页的序号。通过该事件可对激活页进行初始化操作。

18.11.3 多页实例——报名登记

本例使用多页控件创建一个【报名登记】用户窗体，该窗体由多页控件分为 2 页，如图 18-15 所示。

在【报名登记】用户窗体中，没有对多页控件进行编程，只是将多页控件作为数据分类的容器。多页控件只需使用默认属性值即可。其他控件属性的设置也不再列出，下面列出【保存】按钮和【关闭】按钮的相关代码。



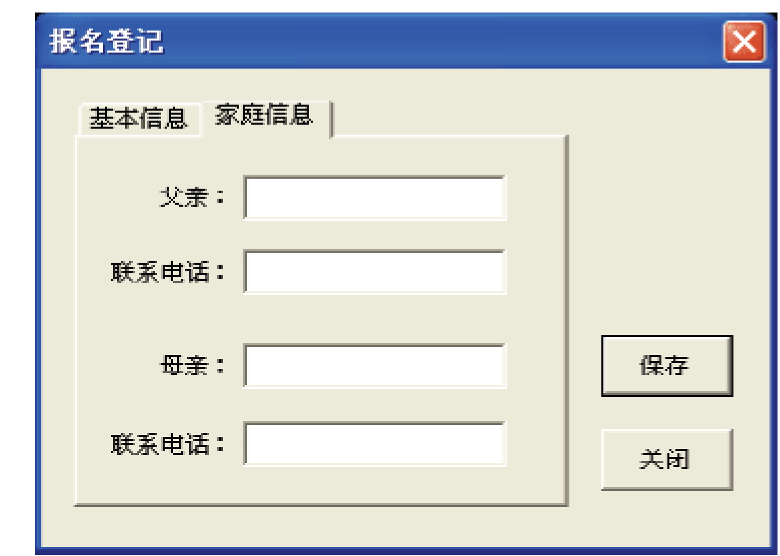


图 18-15 【报名登记】对话框

（1）单击【保存】按钮，将分别从多页控件的两页中放置的控件里获取数据，然后将其保存到工作表中即可，具体代码如下：

```
Private Sub cmdSave_Click()
    Dim r As Long
    If txtName.Value = "" Then
        MsgBox "请输入学员姓名! ", vbCritical + vbOKOnly
        Exit Sub
    End If
    With Worksheets("花名册")
        r = .Range("A1").End(xlDown).Row + 1     '找到添加数据的行
        .Cells(r, 1) = txtName.Value
        If optMan.Value Then
            .Cells(r, 2) = "男"
        Else
            .Cells(r, 2) = "女"
        End If
        .Cells(r, 3) = txtAge.Value              '年龄
        .Cells(r, 4) = txtAddress.Value          '家庭住址
        .Cells(r, 5) = txtFName.Value            '父亲姓名
        .Cells(r, 6) = txtFP.Value               '父亲电话
        .Cells(r, 7) = txtMName.Value            '母亲姓名
        .Cells(r, 8) = txtMP.Value               '母亲电话
    End With
End Sub
```

（2）【关闭】按钮的 VBA 代码如下：

```
Private Sub cmdClose_Click()
    Unload Me
End Sub
```

18.12　RefEdit

使用 RefEdit 控件可折叠当前操作的对话框，让用户在 Excel 工作表中拖动鼠标选择一个单元格区域。

RefEdit 控件的使用很简单，将其放置在窗体的合适位置即可。

18.12.1　RefEdit 常用属性

RefEdit 控件最主要的属性就是 Value，通过该属性可返回用户选择的绝对引用单元格地址字符串，如“Sheet1!$A$1:$F$15”，VBA 程序中获取该字符串，以进行后续的操作。

18.12.2　RefEdit 实例——设置单元格格式

本例制作如图 18-16 所示的窗体，在用户窗体上方设置一个 RefEdit 控件，用来选择需要设置格式的单元格区域。在窗体中设置字体、字形、字号等参数，在【预览】部分将显示设置的效果。单击【设置】按钮，即可对选择的单元格区域设置相应的格式。

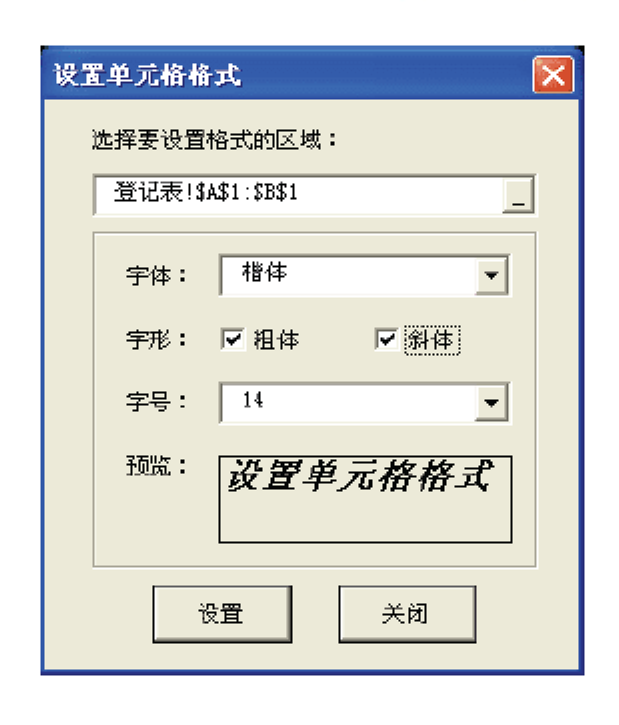


图 18-16　【设置单元格格式】对话框

制作如图 18-16 所示用户窗体的具体步骤如下：

（1）在 VBE 中新增一个用户窗体，向窗体中增加 1 个 Ref Edit 控件、1 个框架、7 个标签、2 个复合框、2 个复选框和 2 个按钮。调整各控件的布局如图 18-16 所示。各控件的属性设置比较简单，这里就不再列出来。

（2）在窗体的初始化事件中编写代码，向【字体】和【字号】复合框中添加相应的选项，并设置两个复选框的初始值。

```
Private Sub UserForm_Initialize()
    With cmbFont
        .AddItem "宋体"
        .AddItem "黑体"
        .AddItem "楷体"
        .AddItem "仿宋_GB2312"
        .Value = .List(0)
    End With

    For i = 8 To 72
        cmbSize.AddItem i
    Next
```

```
    cmbSize.Value = 8
End Sub
```

（3）为了能在名为 lblPreview 的标签控件中查看预览效果，需要在每个设置字体格式控件的 Change 事件中编写代码，改变 lblPreview 标签的字体样式。具体代码如下：

```
Private Sub chkBold_Click()        '预览字形(粗体)
    lblPreview.Font.Bold = chkBold.Value
End Sub

Private Sub chkItalic_Click()      '预览字形(斜体)
    lblPreview.Font.Italic = chkItalic.Value
End Sub

Private Sub cmbFont_Change()       '预览字体
    lblPreview.Font.Name = cmbFont.Value
End Sub

Private Sub cmbSize_Change()       '预览字号
    lblPreview.Font.Size = cmbSize.Value
End Sub
```

（4）单击【设置】按钮，将对选中的区域设置指定的格式，具体代码如下：

```
Private Sub cmdSet_Click()
    Dim rng As Range, i As Integer, str1 As String
    If RefEdit1.Value = "" Then           '未选择单元格区域
        MsgBox "请先在工作表区域拖动鼠标，选择一个单元格区域！", vbCritical +
        vbOKOnly
        Exit Sub
    End If

    i = InStr(1, RefEdit1.Value, "!")    '去掉工作表名
    str1 = Mid(RefEdit1.Value, i + 1)
    Set rng = Range(str1)                 '获取对单元格区域的引用

     With rng.Font                        '设置格式
        .Name = cmbFont.Value             '字体
        .Size = cmbSize.Value             '字号
        .Bold = chkBold.Value             '粗体
        .Italic = chkItalic.Value         '斜体
     End With
End Sub
```

第 19 章　使用 ActiveX 控件

使用 VBA 提供的标准控件可满足大多数应用程序的需要。但对一些特定的需求（或为了简化应用程序的开发），可在应用程序中使用 ActiveX 控件。ActiveX 是由用户使用程序设计语言定制的可重用对象，在 Windows XP 系统中自带许多这类控件。本章介绍几个常用 ActiveX 控件的使用方法。

19.1　添加 ActiveX 控件

本节首先简单介绍 ActiveX 控件的概念，接着介绍将 ActiveX 控件添加到【工具箱】窗口中的步骤。

19.1.1　什么是 ActiveX 控件

Microsoft ActiveX 控件以前也叫做 OLE 控件或 OCX 控件，它是采用 COM 技术创建的可重用的小对象，可以将其插入到其他应用程序中。

ActiveX 控件是可选的控件，它可以被添加到工具箱中并在窗体里使用。当安装 VB 开发环境或其他应用程序时，这些安装程序可能会将其安装包内的 ActiveX 控件文件复制到相应的文件夹中（扩展名为“.ocx”）。

可以使用各种编程语言开发 ActiveX 控件，例如 VB、VC、Java 等。ActiveX 控件一旦被开发出来，设计和开发人员就可以把其当作预装配组件，用于开发应用程序。在使用 ActiveX 控件时，使用者不需要知道这些组件是如何开发的，在很多情况下，甚至不需要自己编程，就可以完成应用程序的设计。

19.1.2　添加 ActiveX 控件到工具箱

默认情况下，【工具箱】中将只显示标准控件。若需要在用户窗体中使用 ActiveX 控件，则需要首先将 ActiveX 控件添加到【工具箱】中。

若要在用户窗体中使用进度条控件，则需要按以下步骤将其添加到【工具箱】中：

（1）在 VBE 环境中，执行主菜单【插入】|【用户窗体】命令，在工程中增加一个用户窗体，同时显示出【工具箱】。

提示：若【工具箱】浮动窗口未显示出来，需执行主菜单【视图】|【工具箱】命令将其显示出来。

（2）右击【工具箱】的空白位置，将弹出如图 19-1 所示的菜单。

（3）单击弹出菜单中的“附加控件”菜单，打开如图 19-2 所示【附加控件】对话框。在该对话框中，可以将控件或对象（例如 Word 文档）添加到工具箱中。

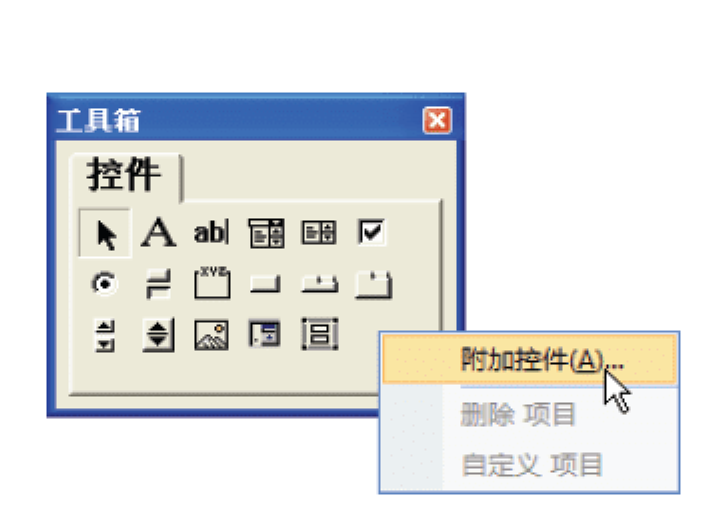






图 19-1　快捷菜单　　　　图 19-2　【附加控件】对话框

（4）在【可用控件】列表框中单击 Microsoft ProgressBar Control 6.0 (SP6)复选框，如图 19-3 所示。

（5）单击【确定】按钮后，【工具箱】中将新增加控件 ProgressBar，如图 19-4 所示。

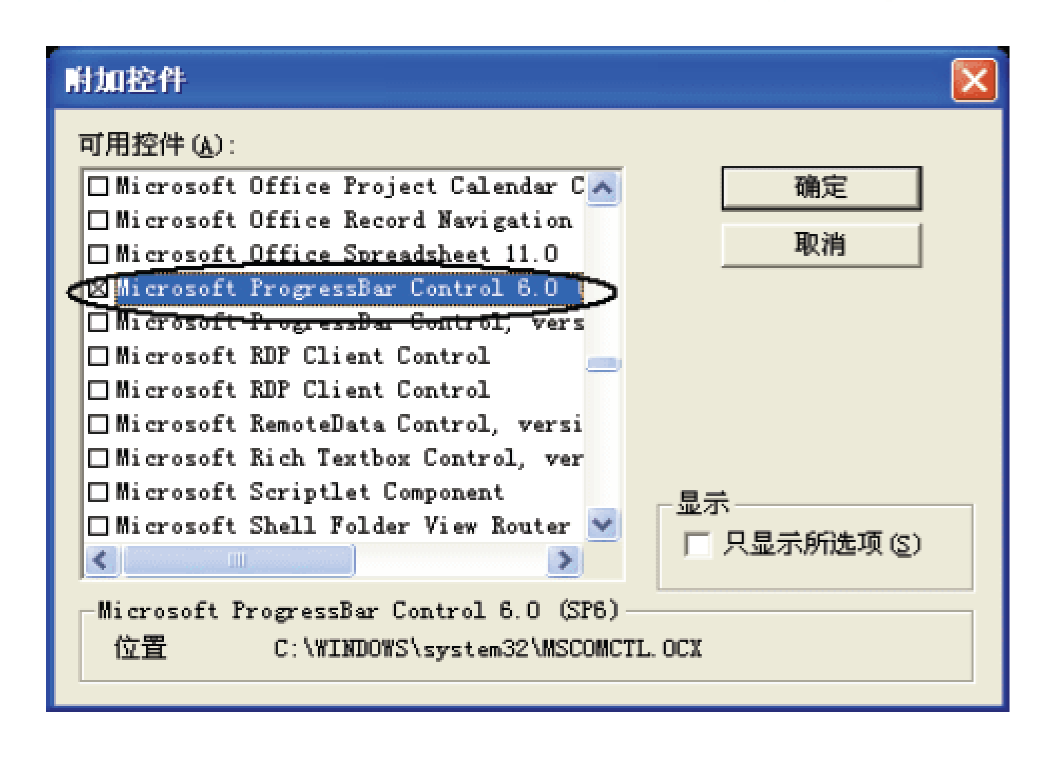




图 19-3　选中 ProgressBar 控件　　　　图 19-4　添加的控件

将 ActiveX 控件添加到工具箱中以后，可与使用标准控件一样的方法，向窗体中添加该图像列表控件。

（6）右击【工具箱】中新增加的控件按钮，打开如图 19-5 所示的快捷菜单，选择【删除 ProgressBar】命令可将该 ActiveX 控件从【工具箱】中删除。

（7）在图 19-5 所示的快捷菜单中选择【自定义 ProgressBar】命令，将打开如图 19-6 所示的对话框，在该对话框中可修改工具按钮的提示文字，还可修改按钮的显示图标。

注意：每个 ActiveX 控件对应一个扩展名为“.OCX”的文件（一个 OCX 文件也可包含多个控件）。




图 19-5　快捷菜单

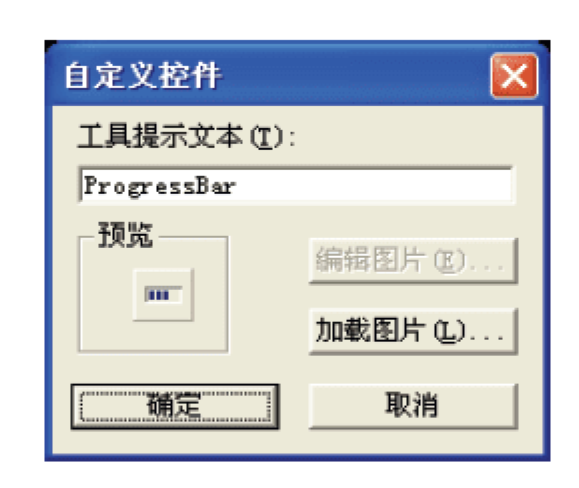


图 19-6 【自定义控件】对话框

19.2　使用进度条控件

Microsoft 公司提供了一个进度条控件，在需要计算机进行长时间运算时，使用进度条显示运算的进度，对用户起到提示作用。在 19.1 节中已经演示了将其添加到【工具箱】中的方法。

19.2.1　进度条控件的常用属性

ActiveX 控件也具有控件的共有属性，例如 Name、Enabled、Visible、Height、Width 等。ProgressBar 控件除了控件的共有属性外，最常用的还有以下 3 个属性。

- ❑ Max：最大值；
- ❑ Min：最小值；
- ❑ Value：进度条的当前值。

ProgressBar 控件根据 Value 和 Max、Min 之间的关系来决定进度条的位置。Max、Min 的默认值为 100 和 0。最简单的情况是，将正在处理的起始值设置到 Min 属性中，将处理的结束值设置到 Max 属性中，而当前处理的值设置到 Value 属性中。

19.2.2　进度条控件的方法

进度条控件有两个方法，分别如下所述。

- ❑ OLEDrag：用于处理拖动进度条的操作。在一般窗体中不使用。
- ❑ Refresh 方法：按设置的 Value 值更新进度条的指示。一般也不显式调用，当设置了进度条控件的 Value 属性值时，进度条指示会自动更新。

19.2.3　进度条实例——隐藏行

下面用实例演示 ActiveX 进度条控件的使用方法。

Excel 工作表的行很多，要隐藏偶数行需要执行很长时间。这时，可使用进度条控件

让用户了解程序执行的进度。

（1）在 VBE 中，执行主菜单【插入】|【用户窗体】命令，向工程中增加一个用户窗体。

（2）使用 19.1 节介绍的方法将进度条控件添加到【工具箱】中。

（3）向用户窗体中添加 1 个命令按钮、1 个框架，在框架中添加 1 个标签控件、1 个进度条控件。调整各控件的布局如图 19-7 所示，设置各控件的属性如表 19-1 所示。

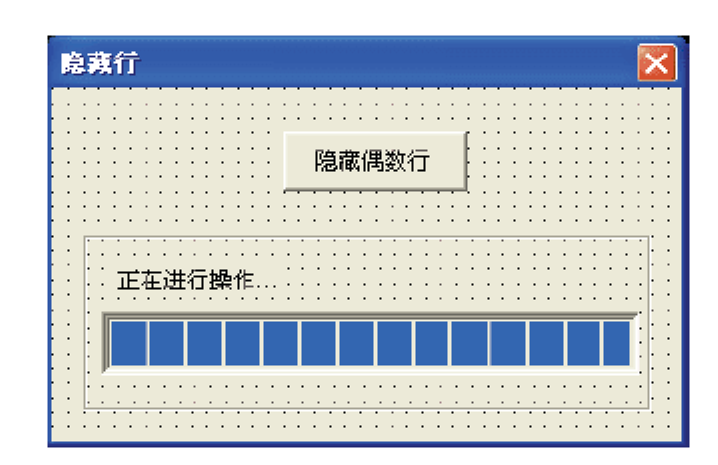


图 19-7　用户窗体

表 19-1　控件属性

用　　途	对　　象	属　　性	属　性　值
主窗体	窗体	名称	frmProgress
		Caption	隐藏行
		Height	168
		Width	250
隐藏行	按钮	名称	cmdHide
		Caption	隐藏偶数行
进度条	进度条	名称	Pb1

（4）给窗体在初始化事件过程（Initialize）中编写以下代码，设置用户窗体的高度，将窗体的高度缩小，只显示上方的按钮。同时隐藏框架控件（框架控件中的控件也将同时隐藏）。

```
Private Sub UserForm_Initialize()
    Me.Height = 83
    Frame1.Visible = False  '隐藏框架及其内部控件
End Sub
```

（5）给【隐藏偶数行】按钮编写代码，用来隐藏工作表 Sheet2 中的偶数行，并更新进度条中的显示进度块。具体代码如下：

```
Private Sub cmdHide_Click()
    Dim r As Long
    r = worksheets("Sheet2").Rows.Count
    Me.Height = 168
    Frame1.Visible = True           '显示框架及其内部控件
    pb1.Min = 0
    pb1.Max = r
```

```
    pb1.Value = 0

    With Worksheets("Sheet2")
    For i = 1 To r
        If i Mod 2 = 0 Then
            .Rows(i).Hidden = True    '隐藏行
        End If
        pb1.Value = I                 '更新进度条
        DoEvents                      '转让控制权
    Next
    End With
    Me.Height = 83
End Sub
```

（6）执行主菜单【插入】|【模块】命令，向工程中增加一个模块，编写以下代码显示用户窗体：

```
Sub 显示进度条()
    frmProgress.Show
End Sub
```

（7）在工作表 Sheet1 中增加一个按钮，将宏【显示进度条】指定给该按钮，设置该按钮显示的文字为“显示进度条”。

（8）单击工作表 Sheet1 中的【显示进度条】按钮，将打开如图 19-8 所示的用户窗体。单击【隐藏偶数行】按钮，程序在隐藏行的同时，将在用户窗体下方显示处理的进度，如图 19-9 所示。



图 19-8　初始状态



图 19-9　进度条

19.3　使用图像列表控件

图像列表控件 ImageList 包含一个图像的集合，这些图像可被其他需要使用图像的控件使用，例如，标准控件中的 Image、按钮等控件，以及 TreeView 控件、ListView 控件等。

19.3.1　图像列表控件简介

ImageList 控件像图像的储藏室，可用来保存窗体中要用到的图像。在程序运行时该控

件是隐藏的，其他控件通过索引来调用该控件中保存的图像。

ImageList 控件包含一个 ListImages 集合，该集合由 ListImage 对象构成。ListImage 对象是任意大小的图片，可以在其他控件中使用。图片可以是位图（.bmp）、光标（.cur）、图标（.ico）、JPEG（.jpg）或 GIF（.gif）文件。

用 ImageList 控件存储图像可以节约程序的开发时间，因为这样可以使编写的代码引用单一和一致的图像目录，而不用在每次显示图片时都使用 LoadPicture 函数从磁盘上装载图片。用户只需要使用一次 LoadPicture 函数将图片填充到 ImageList 控件中，并分配引用的 Key 值，这样在后续的代码中只需要根据 Key 或 Index 属性来引用 ImageList 控件中存储的图像即可。

ImageList 控件全名为 Microsoft ImageList Control 6.0，使用之前必须先通过【附加控件】将其附加到工具箱中，如图 19-10 所示。具体操作参见本章 19.1 节中的内容。

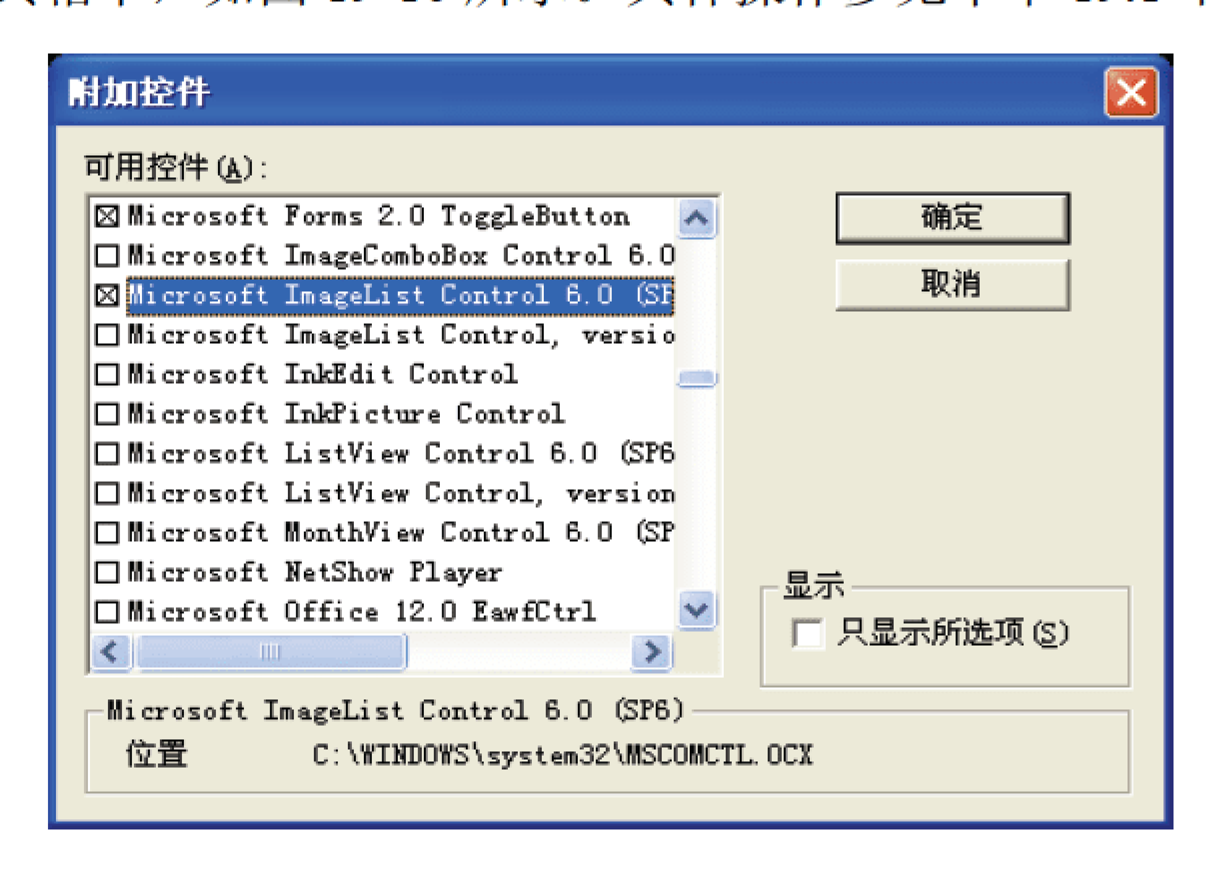


图 19-10　附加 ImageList 控件

19.3.2　图像列表控件的属性

作为图像的存储控件，ImageList 控件的属性很少，主要包括以下几项：

- ImageHeight 属性：该属性返回或设置图像列表控件中 ListImage 对象（图片）的高度。
- ImageWidth 属性：该属性返回或设置图像列表控件中 ListImage 对象（图片）的宽度。
- ListImages 属性：该属性返回对图像列表控件中 ListImage 对象集合的引用。

19.3.3　图像列表控件的方法

ImageList 控件提供了一个方法，即 Overlay 方法。使用该方法可将两个图像组合放置在图像控件 Image 中，该方法的语法格式如下：

```
ImageList.Overlay(Key1, Key2)
```

Overlay 方法的第一个参数代表的图像在下面，第二个参数代表的图像在上面。

ImageList 控件使用一个 ListImage 对象集合保存图片，可使用标准的集合方法（例如，Add 和 Clear 方法）来操作 ListImage 对象。集合中的每一个成员都可以通过其索引或唯一关键字来访问。当把 ListImage 对象添加到一个集合中时，这些索引或唯一关键字将被分别存储在 Index 和 Key 属性中。

19.3.4 添加图像到 ImageList 控件

有两种方法将图像添加到 ImageList 控件中，一是在程序中编写代码，使用 ListImages 集合对象的 Add 方法将图像添加到该集合中，其操作方法与其他集合对象相同。另一种是使用可视的方法，在设计阶段将图像载入控件中。

1. 可视方式添加图像

具体操作步骤如下：

（1）在如图 19-11 所示的 ImageList 控件的【属性】窗口中可单击【(自定义)】选项后的...按钮，打开【属性】对话框，如图 19-12 所示。



图 19-11 【属性】对话框

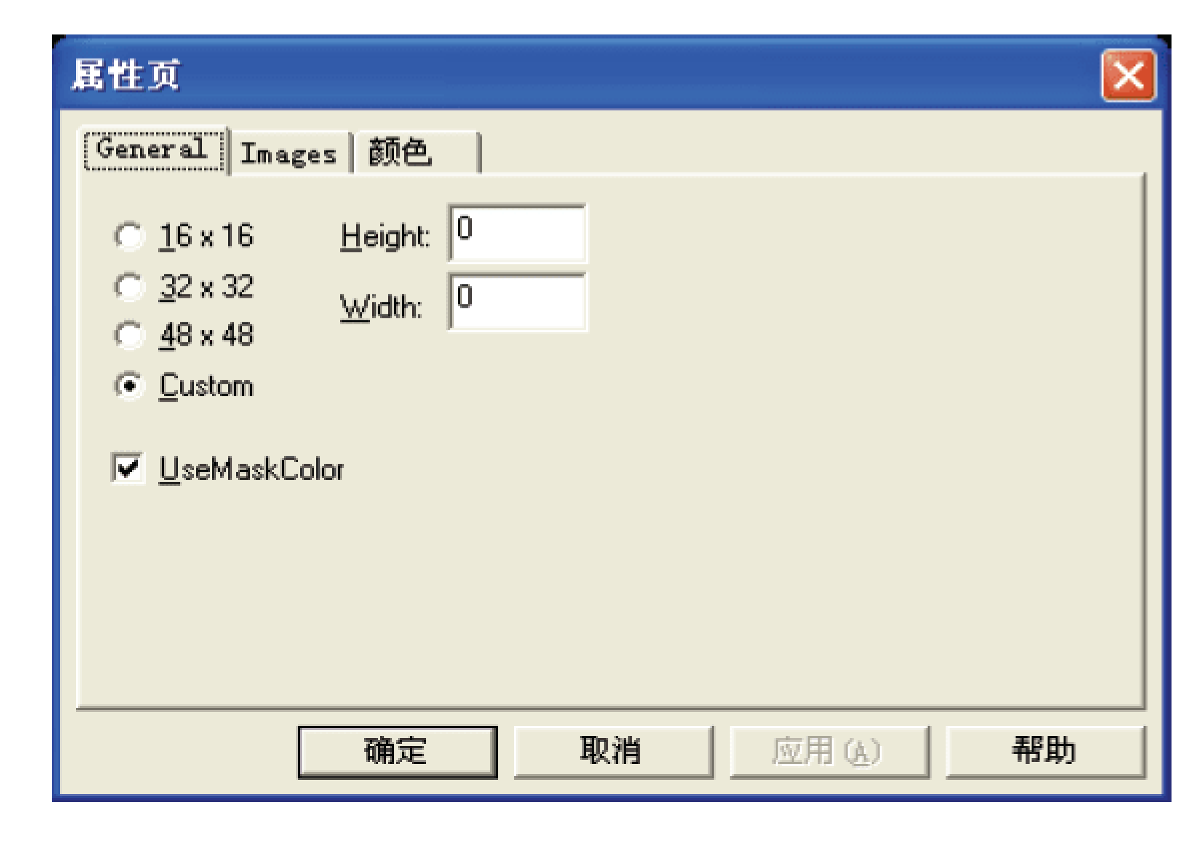


图 19-12 General 选项卡

（2）在 General 选项卡中，可设置图像的大小等参数。

（3）单击 Images 标签，打开如图 19-13 所示的选项卡。在该选项卡中将显示控件内存储的图像。

（4）单击 InsertPicture 按钮，打开 Select Picture 对话框，如图 19-14 所示。

（5）选择需要添加到 ImageList 控件的 ListImages 集合中的图像（可一次选中多个图像文件），单击【打开】按钮即可。

（6）增加图像后，可在图 19-15 所示的对话框中单击选择一个图像，然后修改其 Key 和 Tag 等参数。也可单击 Remove Picture 按钮删除选中的图像。

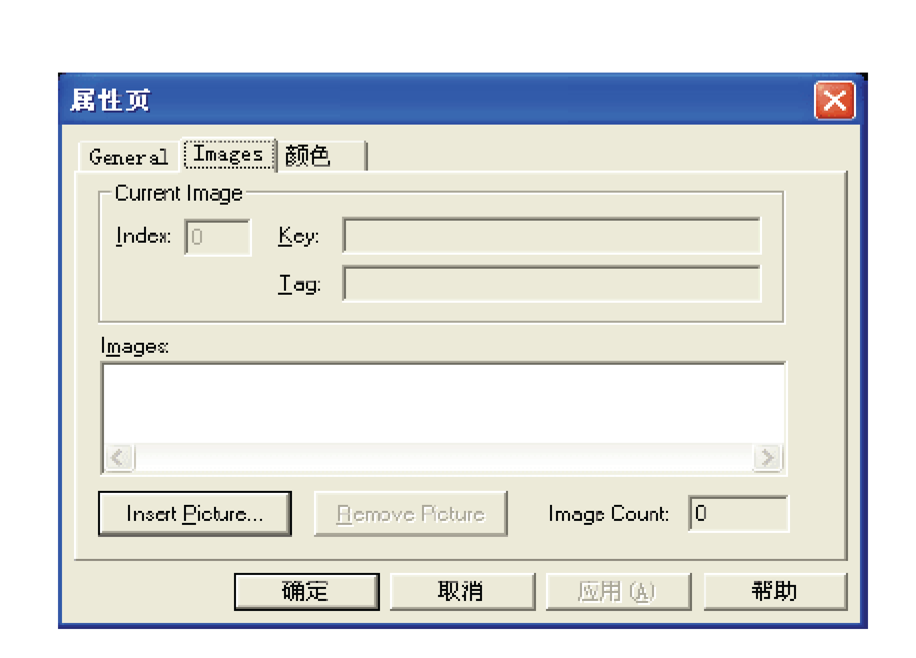


图 19-13　Images 选项卡

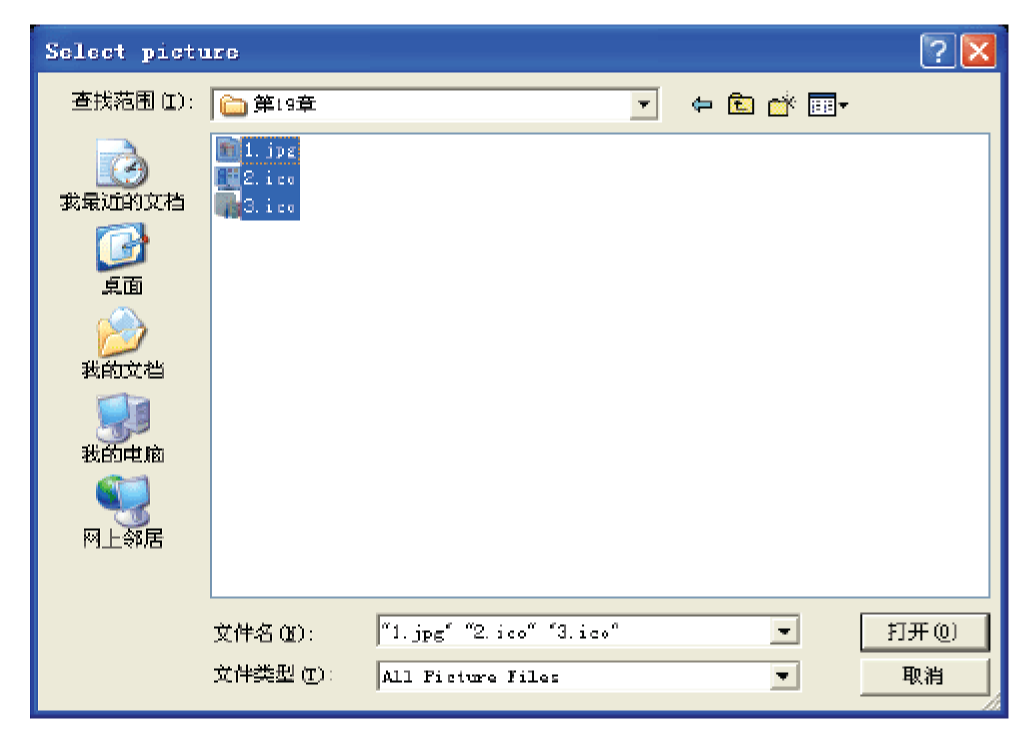


图 19-14　Select Picture 对话框

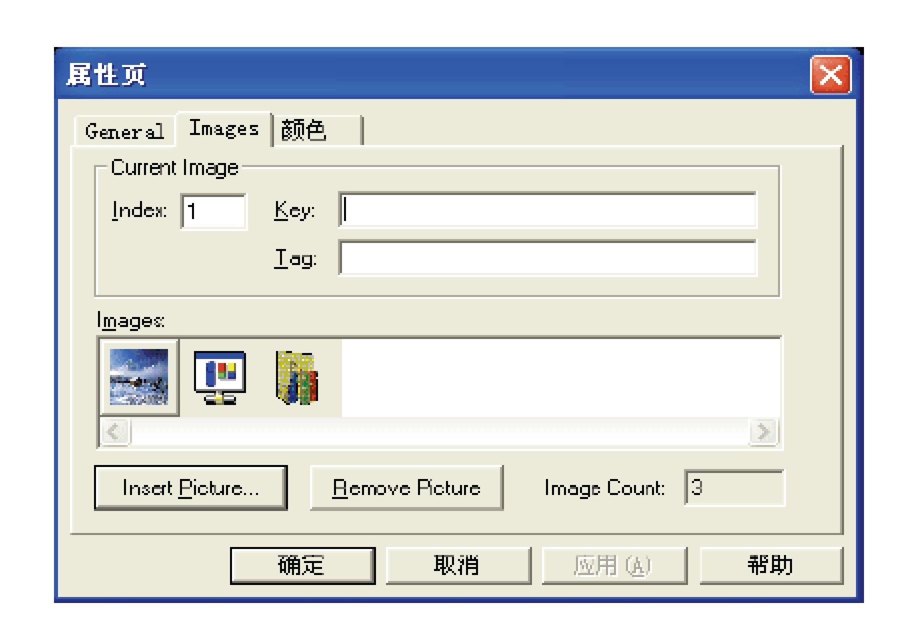


图 19-15　【属于页】对话框

2．使用代码添加图像

在程序运行过程中，还可以使用以下 VBA 代码向 ImageList 控件中添加图像：

```
Dim sPath As String
sPath = ThisWorkbook.Path & "\"
With ImageList1.ListImages
    .Add , "a", LoadPicture(sPath & "1.jpg")
    .Add , "b", LoadPicture(sPath & "2.ico")
    .Add , "c", LoadPicture(sPath & "3.ico")
End With
```

以上代码可将当前工作簿所在目录的图片装入 ImageList 控件中。

19.3.5　图像列表控件实例

下面通过实例演示 ImageList 控件的使用方法。

（1）在 VBE 环境中，执行主菜单【插入】|【用户窗体】命令增加一个用户窗体。

（2）参见本章 19.1 节的方法，向【工具箱】窗口中添加 ImageList 控件。

（3）向用户窗体中添加 1 个图像控件、3 个命令按钮控件、1 个 ImageList 控件。设置各控件的布局如图 19-16 所示，并设置各控件的属性值如表 19-2 所示。

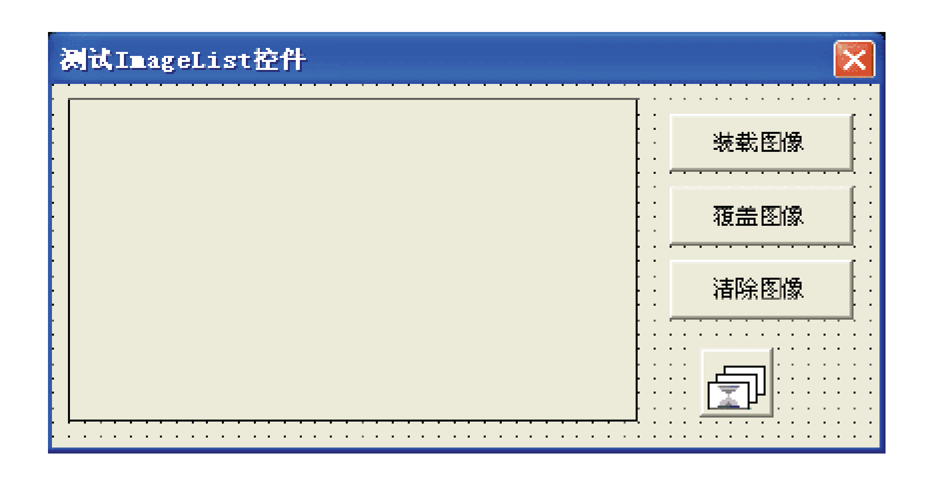


图 19-16　测试 ImageList 控件

提示：因 ImageList 控件在运行时不会显示，所以可将其放置在窗体的任何位置。

表 19-2　控件属性

用　途	对　象	属　性	属　性　值
主窗体	窗体	名称	frmImageList
		Caption	测试ImageList控件
图像列表	图像列表控件	名称	ImageList1
显示图像	图像	名称	Image1
载入图片	命令按钮1	名称	cmdLoad
		Caption	装载图像
组合图片	命令按钮2	名称	cmdOver
		Caption	覆盖图像
清除图片	命令按钮3	名称	cmdClear
		Caption	清除图像

（4）在用户窗体的初始化事件中编写以下代码，向 ImageList 控件中添加图像：

```
Private Sub UserForm_Initialize()
    Dim sPath As String
    sPath = ThisWorkbook.Path & "\"
    With ImageList1.ListImages
        .Add , "a", LoadPicture(sPath & "1.jpg")
        .Add , "b", LoadPicture(sPath & "2.ico")
        .Add , "c", LoadPicture(sPath & "3.ico")
        .Add , "d", LoadPicture(sPath & "4.jpg")
    End With
End Sub
```

（5）【装载图像】按钮用于将 ImageList 控件中的图像显示到 Image 控件中。可以设置 Image 控件的 Picture 属性为 ImageList 控件中的某一个图像即可，具体代码如下：

```
Private Sub cmdLoad_Click()
    Set Image1.Picture = ImageList1.ListImages("a").Picture
End Sub
```

（6）【覆盖图像】按钮用于使用 ImageList 控件的 Overlay 方法，将 ImageList 控件中

的两个图片显示到 Image 控件中，具体代码如下：

```
Private Sub cmdOver_Click()
    ImageList1.MaskColor = vbWhite  '屏蔽颜色为白色
    Set Image1.Picture = ImageList1.Overlay("a", "d")
End Sub
```

（7）【清除图像】按钮清除 Image 控件中的图像，具体代码如下：

```
Private Sub cmdClear_Click()
    Set Image1.Picture = LoadPicture("")
End Sub
```

（8）在“模块 1”中编写以下子过程，用来显示前面创建的窗体：

```
Sub 测试 ImageList 控件()
    frmImageList.Show
End Sub
```

（9）在工作表 Sheet1 中增加一个按钮，将宏【测试 ImageList 控件】指定给该按钮，并设置该按钮显示的文字为图像列表控件。

（10）单击工作表 Sheet1 中的【图像列表控件】按钮，将打开如图 19-17 所示的用户窗体。

（11）单击【装载图像】按钮，左侧图像控件中将显示一个图像，如图 19-18 所示。

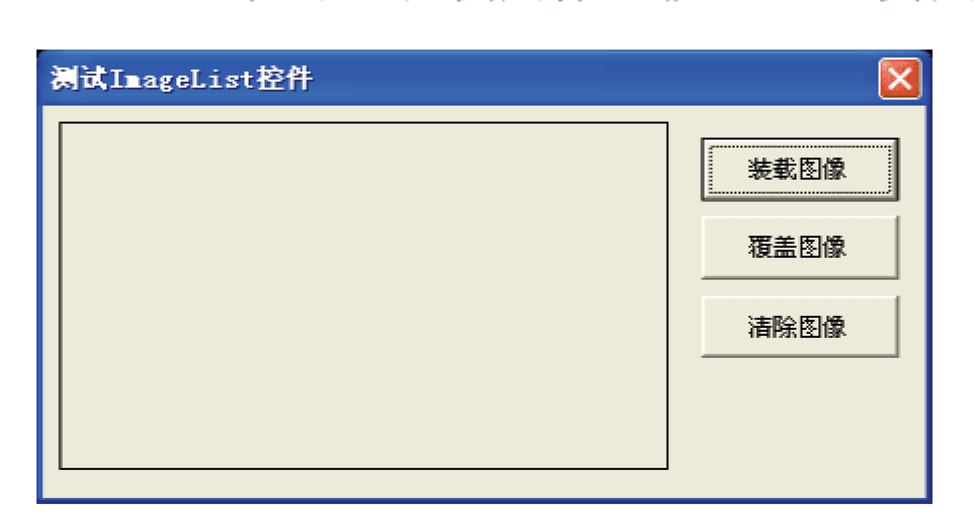


图 19-17 【图像列表控件】对话框



图 19-18 【装载图像】选项的显示

（12）单击【覆盖图像】按钮，在已有图像上方将覆盖显示文字，如图 19-19 所示。



图 19-19 【覆盖图像】选项的显示

（13）单击【清除图像】按钮，图像控件中图像将被清除，得到如图 19-17 所示效果。

19.4 使用树形视图控件

树形视图控件（TreeView）是很有用的一个 ActiveX 控件，一般用于显示文档标题、索引入口、磁盘上的文件和目录、或能被有效地分层显示的各种信息。

19.4.1 树形视图控件简介

Windows 的资源管理器左侧的文件夹列表，就是用 TreeView 控件的典型应用，如图 19-20 所示。各盘符之前有一个加号图标，单击它将展开显示该磁盘包含的文件夹。当展开数据项后，加号图标将变为减号图标，单击减号图标可将展开的内容折叠。

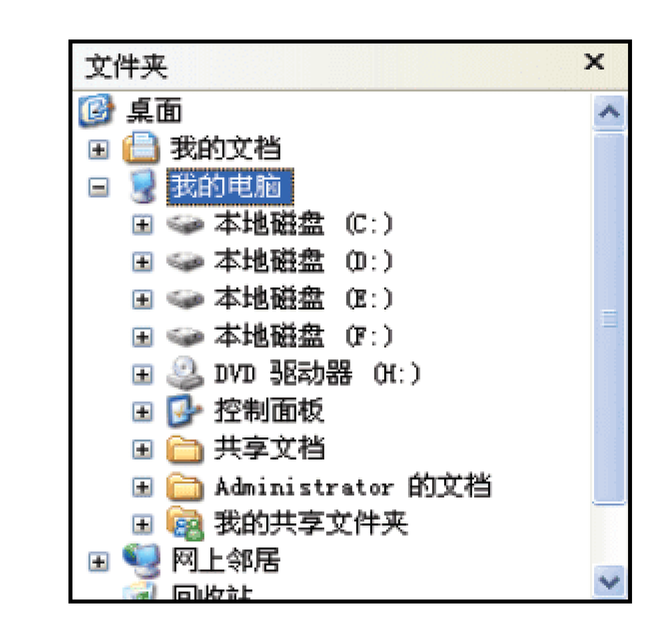


图 19-20 TreeView 控件的应用

TreeView 控件显示的是一个分层列表，每个列表项为一个 Node 对象，每个 Node 对象均由一个标签和一个可选的图像组成。

创建了 TreeView 控件之后，可以通过设置属性与调用方法对各 Node 对象进行操作，这些操作包括添加、删除、对齐和其他操作。可以通过编程展开与折回 Node 对象来显示或隐藏所有子节点。

Node 对象使用 Root、Parent、Child、FirstSibling、Next、Previous 和 LastSibling 属性。在代码中可通过检索对 Node 对象的引用，从而在树上定位。也可以使用键盘在 TreeView 控件中进行定位。

TreeView 控件使用由 ImageList 属性指定的 ImageList 控件，来存储显示于 Node 对象的位图和图标。任何时刻，TreeView 控件只能使用一个 ImageList。当 TreeView 控件的 Style 属性被设置成显示图像的样式时，TreeView 控件中每一项的旁边都有一个同样大小的图像。

TreeView 控件全名为 Microsoft TreeView Control 6.0，使用之前必须先通过【附加控件】将其附加到工具箱中，如图 19-21 所示。



图 19-21 附加 TreeView 控件

19.4.2　树形视图控件常用属性

TreeView 控件有很多的属性和方法，通过这些属性和方法使用 TreeView 可以完成很复杂的功能。下面介绍 TreeView 控件常用的属性。

- CheckBoxes 属性：设置在每一项节点的旁边是否显示一个复选框，类似复选框控件的作用。
- Hottracking 属性：设置为 True 时，当鼠标指针经过某个条目，这些条目是否突出显示，类似网页的超链接效果。
- ImageList 属性：设置与 TreeView 控件相关的 ImageList 控件，在 TreeView 控件的节点中使用 ImageList 控件提供的图像。
- LabelEdit 属性：设置一个值，用来确定是否可以编辑在 TreeView 控件中的 Node 对象的标签。
- LineStyle 属性：这个属性设置在 Node 对象之间显示的线的样式，可设置为以下两个值：
  - 0：三线。显示在 Node 相邻节点和它们的父 Node 之间的线。
  - 1：根线。除了显示在 Node 相邻节点和它们的父 Node 之间的线以外，还显示根节点之间的线。
- Nodes 属性：这个属性返回对 TreeView 控件的 Node 对象集合的引用。可以使用标准的集合方法（如：Add 和 Remove 方法）操作 Node 对象，可以按其索引或存储在 Key 属性中的唯一键值来访问集合中的每个元素。
- Style 属性：设置各个 Node 对象显示的映像、文本和线条的类型。可设置为以下的值：
  - 0，仅为文本；
  - 1，为图像和文本；
  - 2，为+/–号和文本；
  - 3，为+/–号、图像和文本；
  - 4，为直线和文本；
  - 5，为直线、图像和文本；
  - 6，为直线、+/–号和文本；
  - 7（默认），为直线、+/–号、图像和文本。
- Sorted 属性：设置 Node 对象的根节点或子节点是否按字母顺序。若设置为 True，则 Node 对象根据它们的 Text 属性按字母顺序排列。其 Text 属性将数字开始的 Node 对象也作为字符串排序，第一个数字确定在排序中的初始位置，后面的数字确定以后的排序。若设置为 False，则 Node 对象不排序。
- Singlesel 属性：表示在树中选择新的条目时，是否展开此条目并收拢前一个条目，当设置为 True，并且当前选中的条目有子项的时候，将把子项展开，并将原来选中的条目折叠。

🔔注意：设置 Sorted 属性为 True 仅对当前 Nodes 集合排序。在 TreeView 控件中添加新的 Node 对象时，必须再次设置 Sorted 属性为 True，以便对添加的 Node 对象排列。

19.4.3　树形视图控件的常用方法

使用 TreeView 的方法可向控件中添加项目、获取项目数量等操作，常用的方法如下：

1. Add方法

使用 Add 方法在 Treeview 控件的 Nodes 集合中添加一个 Node 对象。其语法格式如下：

```
object.Add(relative, relationship, key, text, image, selectedimage)
```

各参数的含义如下所述。

- ❑ relative：可省略，代表已存在的 Node 对象的索引号或键值。
- ❑ relationship：可省略，代表新节点与已存在的节点间的关系，指定 Node 对象的相对位置。可设置为以下值：
  - ➢ 0（tvwFirst）为首节点，该 Node 和在 relative 中被命名的节点位于同一层，并位于所有同层节点之前。
  - ➢ 1（tvwLast）为最后的节点，该 Node 和在 relative 中被命名的节点位于同一层，并位于所有同层节点之后。任何连续添加的节点可能位于最后添加的节点之后。
  - ➢ 2（tvwNext）是默认值，为下一个节点，该 Node 位于在 relative 中被命名的节点之后。
  - ➢ 3（tvwPrevious）为前一个节点，该 Node 位于在 relative 中被命名的节点之前。
  - ➢ 4（tvwChild）是默认值，为子节点。该 Node 为在 relative 中被命名的节点的子节点。
- ❑ key：可省略，是一个唯一的字符串，用于在集合中查找 Node 对象。
- ❑ text：表示 Node 对象显示的字符串。
- ❑ image：可省略，代表一个图像或在 ImageList 控件中图像的索引。
- ❑ selectedimage：可省略，代表一个图像或在 ImageList 控件中图像的索引，在 Node 对象被选中时显示该图像。

🔔注意：如果在 relative 中没有被命名的 Node 对象，则新节点被放在节点顶层的最后位置。

2. GetVisibleCount方法

使用该方法可获得在 TreeView 控件的显示区域中 Node 对象的数量。Node 对象的个数取决于在一个窗口中能固定多少行。总的行数取决于控件的高度和 Font 对象的 Size 属性。可以使用 GetVisibleCount 属性确保可视的最小行数，这样可以精确地访问一个层。如果最小行数是不可见的，可以用 Height 属性重新设置 TreeView 的大小。

3．StartLabelEdit方法

使用该方法，可编写选中节点的文本内容。

19.4.4　树形视图控件常用事件

TreeView 控件可对很多事件作出响应，常用的事件有以下几种。

- Collapse 事件：在 TreeView 控件中的任何 Node 对象被折回时，这个事件便发生。
- Collapse 事件发生在标准的 Click 事件之前。这个事件传送对折回的 Node 对象的引用。
- 产生 Collapse 事件有三种方法：其一，设置 Node 对象的 Expanded 属性为 False；其二，双击 Node 对象；其三，在 TreeView 控件的 Style 属性被设置为包括+/–号图像样式的情况下，单击+/–号图像。
- Expand 事件：在 TreeView 控件中的 Node 对象扩展开时，也就是它的子节点变成可视时，这个事件便发生。Expand 事件发生在 Click 和 DblClick 事件之后。
- NodeClick 事件：在一个 Node 对象被单击时，这个事件便发生。NodeClick 事件发生在标准的 Click 事件之前。

在单击节点对象之外 TreeView 控件的任何部位，标准的 Click 事件发生。当单击某个特定的 Node 对象时，NodeClick 事件发生；NodeClick 事件也返回对特定的 Node 对象的引用，并在下一步操作之前，该引用使这个 Node 对象可用。

19.4.5　树形视图控件实例

本例制作如图 19-22 所示的用户窗体。在该窗体中，左侧的树形视图显示人员的姓名及其相关资料，可单击下方的【展开】、【排序】、【删除】按钮对树形视图进行操作。在右侧输入数据，单击【添加】按钮，可将数据添加到树形视图中。

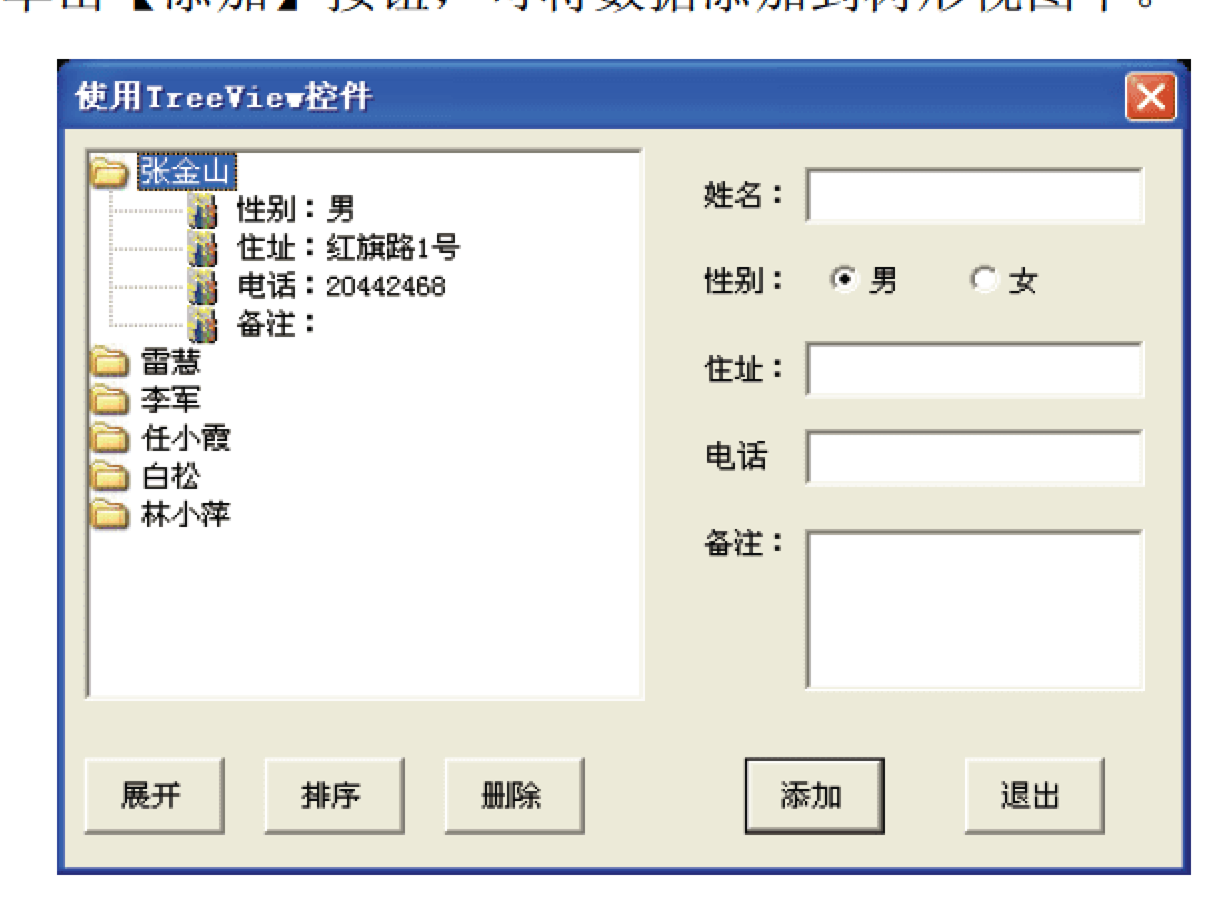


图 19-22 【使用 TreeView 控件】对话框

要制作如图 19-22 所示的用户窗体，可按以下步骤进行：

（1）在 VBE 中插入一个用户窗体。

（2）向【工具箱】中附加 TreeView 控件和 ImageList 控件。

（3）向窗体中添加 1 个 TreeView 控件、1 个 ImageList 控件、5 个标签控件、4 个文字框控件、2 个选项按钮控件、5 个命令按钮控件。调整各控件的布局如图 19-22 所示，设置各控件的属性如表 19-3 所示。

表 19-3　控件属性

用　途	对　象	属　性	属　性　值
主窗体	窗体	名称	frmTreeView
		Caption	测试TreeView控件
图像列表	图像列表控件	名称	ImageList1
树形视图	树形视图控件	名称	TreeView1
姓名	文字框1	名称	txtName
地址	文字框2	名称	txtAddress
电话	文字框3	名称	txtTelephone
备注	文字框4	名称	txtMemo
		MultiLine	True
性别	选项按钮1	名称	optMan
		Caption	男
		Value	True
	选项按钮2	名称	optWoman
		Caption	女
展开	命令按钮1	名称	cmdExpand
		Caption	展开
排序	命令按钮2	名称	cmdSort
		Caption	排序
删除	命令按钮3	名称	cmdDel
		Caption	删除
添加	命令按钮4	名称	cmdAdd
		Caption	添加
		Default	True
退出	命令按钮5	名称	cmdExit
		Caption	退出
		Cancel	True

（4）在用户窗体中选中 ImageList 控件，在【属性】窗口中单击【自定义】右侧的按钮，打开如图 19-23 所示的【属性页】对话框，选中【16×16】选项按钮。

（5）单击切换到 Images 选项卡，然后单击 Insert Picture 按钮增加 3 个图像，分别设置各图像的 Key 为 close、open 和 p，如图 19-24 所示。

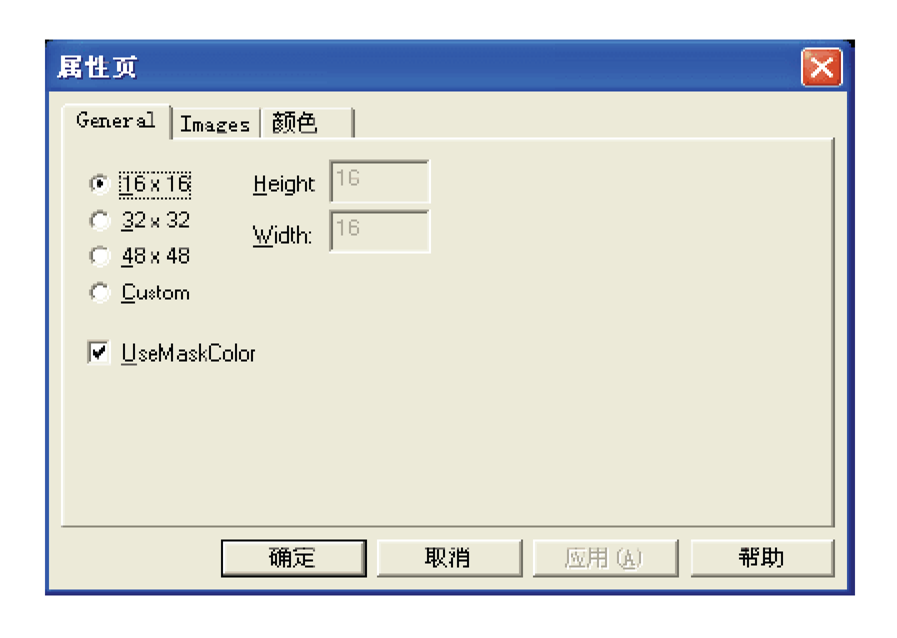


图 19-23　设置 ImageList 控件的属性

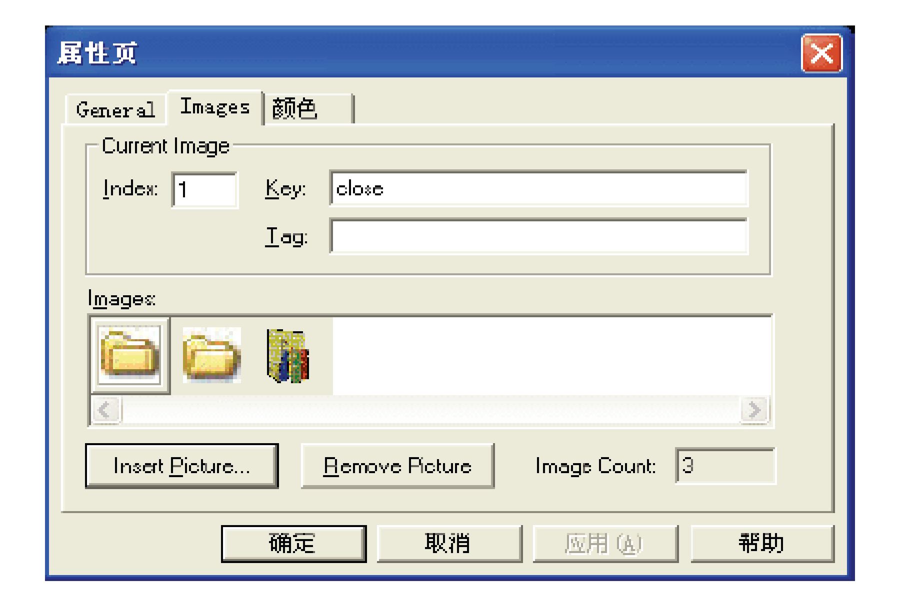


图 19-24　增加图像

（6）在用户窗体的初始化（Initialize）事件中编写代码，从工作表“花名册”中读出数据，添加到 TreeView 控件中，具体的代码如下：

```
Private Sub UserForm_Initialize()
    Dim c As Integer, i As Integer
    Dim nodx As Node
    c = Worksheets("花名册").Range("A1").End(xlDown).Row '数据行数
    If c >= 65536 Then c = 1

    With TreeView1                                    '设置 TreeView 控件的
                                                       属性
        .LineStyle = tvwTreeLines                     '设置线型
        .ImageList = ImageList1                       '绑定 ImageList 控件
        .Style = tvwTreelinesPlusMinusPictureText     '设置各节点的类型
    End With

    i = 2
    With Worksheets("花名册")
    Do While i <= c                                   '逐行读出工作表中的数据
        str1 = .Cells(i, 1)
        Set nodx = TreeView1.Nodes.Add(, , str1, str1, "close", "open")
        Set nodx = TreeView1.Nodes.Add(str1, tvwChild, "sex" & i, "性别: "
        & .Cells(i, 2), "p")
        Set nodx = TreeView1.Nodes.Add(str1, tvwChild, "address" & i, "住
        址: " & .Cells(i, 3), "p")
       Set nodx = TreeView1.Nodes.Add(str1, tvwChild, "telephone" & i,
       "电话: " & .Cells(i, 4), "p")
        Set nodx = TreeView1.Nodes.Add(str1, tvwChild, "memo" & i, "备注: "
        & .Cells(i, 5), "p")
        i = i + 1
    Loop
    End With
End Sub
```

（7）单击【展开】按钮，TreeView 控件中所有节点都将展开，并显示出每一个数据项。同时按钮的文字提示改为“折叠”。单击名为【折叠】的按钮时，TreeView 控件中所有节点都将折叠。该按钮 Click 事件过程的代码如下：

```
Private Sub cmdExpand_Click()
```

```
    Dim i As Integer
    If cmdExpand.Caption = "展开" Then
        For i = 1 To TreeView1.Nodes.Count
            TreeView1.Nodes(i).Expanded = True   '展开所有节点
        Next
        cmdExpand.Caption = "折叠"
    Else
        For i = 1 To TreeView1.Nodes.Count
            TreeView1.Nodes(i).Expanded = False  '折叠所有节点
        Next
        cmdExpand.Caption = "展开"
    End If
End Sub
```

（8）单击【排序】按钮，TreeView 控件中的节点将自动排序。该按钮的 Click 事件代码如下：

```
Private Sub cmdSort_Click()
    TreeView1.Sorted = True
End Sub
```

（9）单击【删除】按钮，将删除 TreeView 控件中选中的节点，具体代码如下：

```
Private Sub cmdDel_Click()
    If TreeView1.SelectedItem.Index <> 1 Then
        TreeView1.Nodes.Remove TreeView1.SelectedItem.Index '删除选定的节点
    End If
End Sub
```

（10）单击【添加】按钮，将右侧输入的内容添加到 TreeView 控件中，具体代码如下：

```
Private Sub cmdAdd_Click()
    Dim c As Integer, i As Integer, str1 As String, strSex As String
    Dim nodx As Node

    str1 = Trim(txtName.Value)
    If Len(str1) = 0 Then                   '判断姓名文字框是否为空
        MsgBox "请输入姓名！"
        Exit Sub
    End If

    c = TreeView1.Nodes.Count               '获取 TreeView 控件中节点数
    For i = 1 To c                          '判断是否已有同名节点
        If TreeView1.Nodes(i).Text = str1 Then
            MsgBox "列表中已经有该姓名，请重新输入！"
            txtName.SetFocus
            Exit Sub
        End If
    Next
    Set nodx = TreeView1.Nodes.Add(, , str1, str1, "close", "open")
                                            '添加节点
    strSex = "男"
```

```
    If optWoman.Value Then strSex = "女"
    Set nodx = TreeView1.Nodes.Add(str1, tvwChild, "sex" & c + 2, "性别: "
    & strSex, "p")
    Set nodx = TreeView1.Nodes.Add(str1, tvwChild, "address" & c + 2, _
        "住址: " & txtAddress.Value, "p")
    Set nodx = TreeView1.Nodes.Add(str1, tvwChild, "telephone" & c + 2, _
        "电话: " & txtTelephone.Value, "p")
    Set nodx = TreeView1.Nodes.Add(str1, tvwChild, "memo" & c + 2, _
        "备注: " & txtMemo.Value, "p")
End Sub
```

（11）单击【退出】按钮卸载窗体，具体代码如下：

```
Private Sub cmdExit_Click()
    Unload Me
End
```

（12）在“模块 1”中编写以下代码，显示用户窗体。

```
Sub 测试 TreeView 控件()
    frmTreeView.Show
End Sub
```

（13）在工作表 Sheet1 中增加一个按钮，将宏【测试 TreeView 控件】指定给该按钮，设置该按钮显示的文字为“树形视图控件”。

（14）单击工作表 Sheet1 中的【树形视图控件】按钮，将打开如图 19-25 所示的用户窗体。在该窗体的 TreeView 控件中已经添加了工作表“花名册”中的数据。

（15）单击【展开】按钮，每个姓名中包含的子节点将展开，同时【展开】按钮的提示文字将改为“折叠”，如图 19-26 所示。




图 19-25　【使用 TreeView 控件】对话框



图 19-26　展开节点

（16）在图 19-26 中单击【折叠】按钮，各节点展开的子节点将折叠，显示为如图 19-25 所示的效果。

（17）单击【排序】按钮，节点数据进行排序，将得到如图 19-27 所示的效果。

（18）在 TreeView 控件中单击选中一个节点，单击【删除】按钮，可删除选中的节点。

（19）在用户窗体右侧输入内容，如图 19-28 所示。

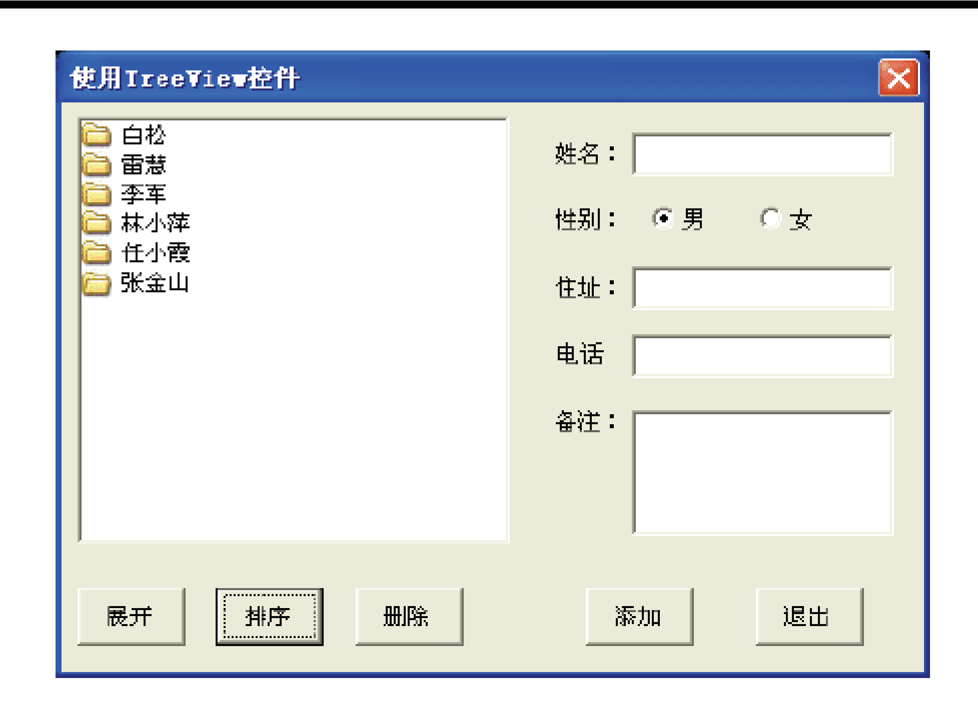


图 19-27　排序的效果



图 19-28　输入数据

（20）输入数据后，单击【添加】按钮即可将输入的内容添加到 TreeView 控件中。单击新添加的节点，将展开该节点，如图 19-29 所示。



图 19-29　添加节点

19.5　使用列表视图控件

列表视图（ListView）控件可使用 4 种不同视图来显示项目。通过此控件，可将项目组成带有或不带有列标头的列，并显示伴随的图标和文本。

19.5.1　列表视图简介

ListView 控件可以用来显示各项带图标的列表，也可以用来显示带有子项的列表，Windows 操作系统的资源管理器中文件夹窗口就是最好的应用例子。

ListView 控件由 ColumnHeader 对象和 ListItem 对象组成，ListView 控件中显示的项目包含在 ListImages 集合中，每个项目为一个 ListItem 对象。使用 ListView 控件可将称作 ListItem 对象的列表条目按以下 4 种视图方式之一显示：

- ❑ 大图标；
- ❑ 小图标；
- ❑ 列表；
- ❑ 报表（详细资料）。

ColumnHeader 对象的个数决定了控件的列数，而 ListItem 对象的个数则决定了控件的行数，如图 19-30 所示。

注意：只有设置为【报表（详细资料）】选项，ListView 才使用 ColumnHeader 对象按行列显示列表框，其他三种模式下只显示 ListItem 对象的名称信息。

ListView 控件全名为 Microsoft ListView Control 6.0，使用之前必须先通过【附加控件】将其附加到工具箱中，如图 19-31 所示。

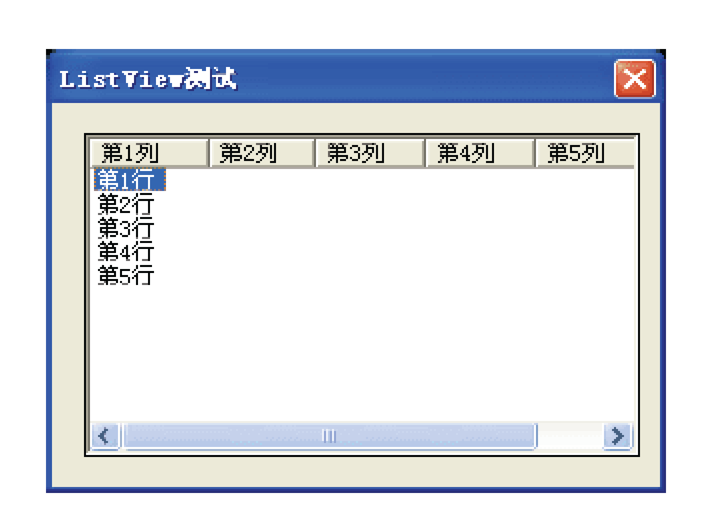


图 19-30　ListView 控件的行和列

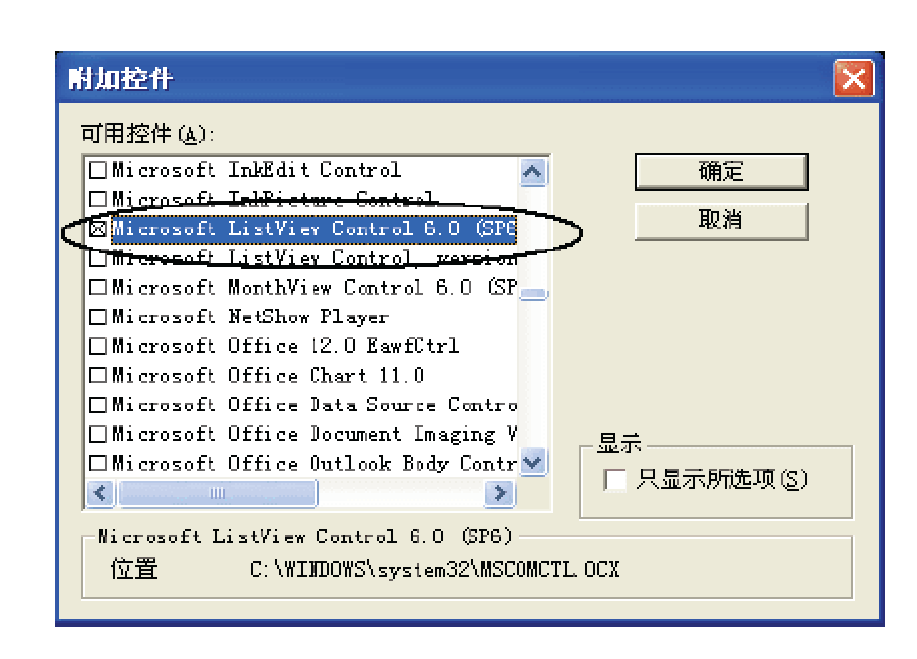


图 19-31　附加 ListView 控件

19.5.2　列表视图控件常用属性

通过 ListView 控件的属性和方法，可以控制显示的项目。ListView 控件常用属性如下所述。

- ❑ Arrange 属性：设置 ListView 控件中的大图标或小图标视图的排列方式。0 表示不排列；1 表示左对齐，项目自动沿控件左侧对齐；2 表示顶对齐，项目自动沿控件顶端对齐。
- ❑ Icons 属性：设置 ListView 控件中大图标（默认）视图时使用的 ImageList 控件。
- ❑ SmallIcons 属性：设置 ListView 控件中小图标视图时使用的 ImageList 控件。
- ❑ MultiSelect 属性：设置用户是否可以选择多个对象或项目。当属性值为 False 时，表示不允许选择多个对象或项目；当属性值为 True 时，则表示允许多个选择。
- ❑ SelectedItem 属性：返回对于一个对象的引用，该对象能用来在选定的对象上设置属性和调用方法。这一属性被典型地用于返回对 ListItem 对象的引用。
- ❑ Sorted 属性：设置集合中的项目是否排序。
- ❑ SortKey 属性：设置 ListView 控件的 ListItem 对象中排序的关键字。
- ❑ SortOrder 属性：设置 ListView 控件的 ListItem 对象以升序或降序排序。0 为升序，1 为降序。

- View 属性：ListView 控件作为一个可以显示图标或者子项的列表控件，它最重要的属性就是 View 属性，该属性决定了以哪种视图模式显示控件的项，可设置为以下 4 种常数。
  - lvwIcon：大图标视图模式，在项的文本旁显示大图标。
  - lvwSmallIcon：小图标视图模式，与大图标模式一样，但是显示的是小图标。
  - lvwList：列表视图模式，显示小图标，但是项是垂直排列的，并且只显示单列。
  - lvwReport：详细资料视图模式，可显示每个 ListItem 项的详细信息。

19.5.3 列表视图控件常用事件

ListView 控件常用事件有以下几个。

- AfterLabelEdit 事件：在编辑当前被选中的 ListItem 对象的标签之后该事件发生。在该事件中编写代码，用来检查标签被修改后的校验工作。当用户单击另一个 ListItem 项目，或者按 Enter 键时，便是编辑操作完成时。要取消标签的编辑操作，可以设置事件过程中的 Cancel 参数为任何非零数或 True。如果标签的编辑操作被取消，则先前存在的标签会被恢复。
- BeforeLabelEdit 事件：在试图编辑当前被选中 ListItem 对象的标签时，产生该事件。在该事件中编写代码，可用来处理编辑标签前的一些准备工作。
- ItemClick 事件：单击 ListView 控件中 ListItem 对象时事件发生，使用该事件决定单击了哪个 ListItem 对象。该事件在 Click 事件之前触发。单击 ListView 控件的任何部分时将触发标准的 Click 事件。只有当单击 ListItem 对象的文本或图像时才触发 ItemClick 事件。

19.5.4 列表视图控件实例

下面使用 ListView 控件显示工作表中的数据，如图 19-32 所示。



图 19-32 用 ListView 控件显示数据

要制作如图 19-32 所示的用户窗体，可按以下步骤进行：

（1）在 VBE 中插入一个用户窗体。

（2）向【工具箱】中附加 ListView 控件和 ImageList 控件。

（3）向窗体中添加 1 个 ListView 控件、2 个 ImageList 控件、6 个标签控件、4 个文字框控件、2 个选项按钮控件、1 个复合框控件、2 个命令按钮控件。调整各控件的布局如图 19-32 所示，设置各控件的属性如表 19-4 所示。

（4）参照 TreeView 控件中的方法，向 ImageList1 控件中添加一个小图标，设置其 Key 为 sIcon。向 ImageList2 控件中添加一个大图标，设置其 Key 为 lIcon。

表 19-4　控件属性

用　　途	对　　象	属　　性	属　性　值
主窗体	窗体	名称	frmListView
		Caption	测试ListView控件
小图标	图像列表1	名称	ImageList1
大图标	图像列表2	名称	ImageList2
列表视图	列表视图控件	名称	ListView1
姓名	文字框1	名称	txtName
地址	文字框2	名称	txtAddress
电话	文字框3	名称	txtTelephone
备注	文字框4	名称	txtMemo
		MultiLine	True
性别	选项按钮1	名称	optMan
		Caption	男
		Value	True
	选项按钮2	名称	optWoman
		Caption	女
视图	复合框	名称	cmbView
添加	命令按钮1	名称	cmdAdd
		Caption	添加
		Default	True
退出	命令按钮2	名称	cmdExit
		Caption	退出
		Cancel	True

（5）在用户窗体的初始化（Initialize）事件中编写代码，设置 ListView 控件的大图标和小图标控件，再从工作表“花名册”中读出数据，添加到 ListView 控件中，具体的代码如下：

```
Private Sub UserForm_Initialize()
    Dim c As Integer, i As Integer
    Dim item As ListItem, colh As ColumnHeader
    c = Worksheets("花名册").Range("A1").End(xlDown).Row      '数据行数
```

```
    If c >= 65536 Then c = 1

    ListView1.Icons = ImageList2              '设置大图标控件
    ListView1.SmallIcons = ImageList1         '设置小图标控件

    With ListView1.ColumnHeaders              '添加列标题
        Set colh = ListView1.ColumnHeaders.Add(, "姓名", "姓名")
        Set colh = ListView1.ColumnHeaders.Add(, "性别", "性别")
        Set colh = ListView1.ColumnHeaders.Add(, "地址", "地址")
        Set colh = ListView1.ColumnHeaders.Add(, "电话", "电话")
        Set colh = ListView1.ColumnHeaders.Add(, "备注", "备注")
    End With

    i = 2
    With Worksheets("花名册")
        Do While i <= c                       '逐行读出工作表中的数据
            str1 = .Cells(i, 1)
            Set item = ListView1.ListItems.Add(, str1, str1, "lIcon", "sIcon")
            item.SubItems(1) = .Cells(i, 2)   '性别
            item.SubItems(2) = .Cells(i, 3)   '地址
            item.SubItems(3) = .Cells(i, 4)   '电话
            item.SubItems(4) = .Cells(i, 5)   '备注
            i = i + 1
        Loop
    End With

    With cmbView                              '初始化视图复合框的值
        .AddItem "大图标"
        .AddItem "小图标"
        .AddItem "列表"
        .AddItem "详细资料"
        .Value = .List(0)                     '以大图标方式显示
    End With
End Sub
```

（6）在【视图】复合框中选择某一个选项后，ListView 控件将按指定的视图方式显示列表项目，该复合框的 Change 事件的代码如下：

```
Private Sub cmbView_Change()
    Select Case cmbView.Value
    Case "大图标"
        ListView1.View = lvwIcon
    Case "小图标"
        ListView1.View = lvwSmallIcon
    Case "列表"
        ListView1.View = lvwList
    Case "详细资料"
        ListView1.View = lvwReport
```

```
    End Select
End Sub
```

（7）单击【添加】按钮，将右侧输入的内容添加到 ListView 控件中，具体代码如下：

```
Private Sub cmdAdd_Click()
    Dim c As Integer, i As Integer, str1 As String, strSex As String
    Dim item As ListItem

    str1 = Trim(txtName.Value)
    If Len(str1) = 0 Then                     '判断姓名文字框是否为空
        MsgBox "请输入姓名！"
        Exit Sub
    End If
    c = ListView1.ListItems.Count             '获取 ListView 控件中的项目数
    For i = 1 To c                            '判断是否已有同名项目
        If ListView1.ListItems(i).Text = str1 Then
            MsgBox "列表中已经有该姓名，请重新输入！"
            txtName.SetFocus
            Exit Sub
        End If
    Next

    Set item = ListView1.ListItems.Add(, str1, str1, "lIcon", "sIcon")
    strSex = "男"
    If optWoman.Value Then strSex = "女"
    item.SubItems(1) = strSex                 '性别
    item.SubItems(2) = txtAddress.Value       '地址
    item.SubItems(3) = txtTelephone.Value     '电话
    item.SubItems(4) = txtMemo.Value          '备注
End Sub
```

（8）单击【退出】按钮卸载窗体，具体代码如下：

```
Private Sub cmdExit_Click()
    Unload Me
End Sub
```

（9）在【模块 1】代码窗口中编写以下代码，显示用户窗体：

```
Sub 测试ListView控件()
    frmListView.Show
End Sub
```

（10）在工作表 Sheet1 中增加一个按钮，将宏【测试 ListView 控件】指定给该按钮，设置该按钮显示的文字为“测试 ListView 控件”。

（11）单击工作表 Sheet1 中的【测试 ListView 控件】按钮，将打开如图 19-33 所示的用户窗体。在该窗体的 ListView 控件中已经添加了工作表“花名册”中的数据。

（12）在【视图】复合框中选择“小图标”，ListView 控件中显示内容如图 19-34 所示。



图 19-33　大图标视图



图 19-34　小图标视图

（13）列表视图方式如图 19-35 所示，详细资料视图如图 19-36 所示。



图 19-35　列表视图

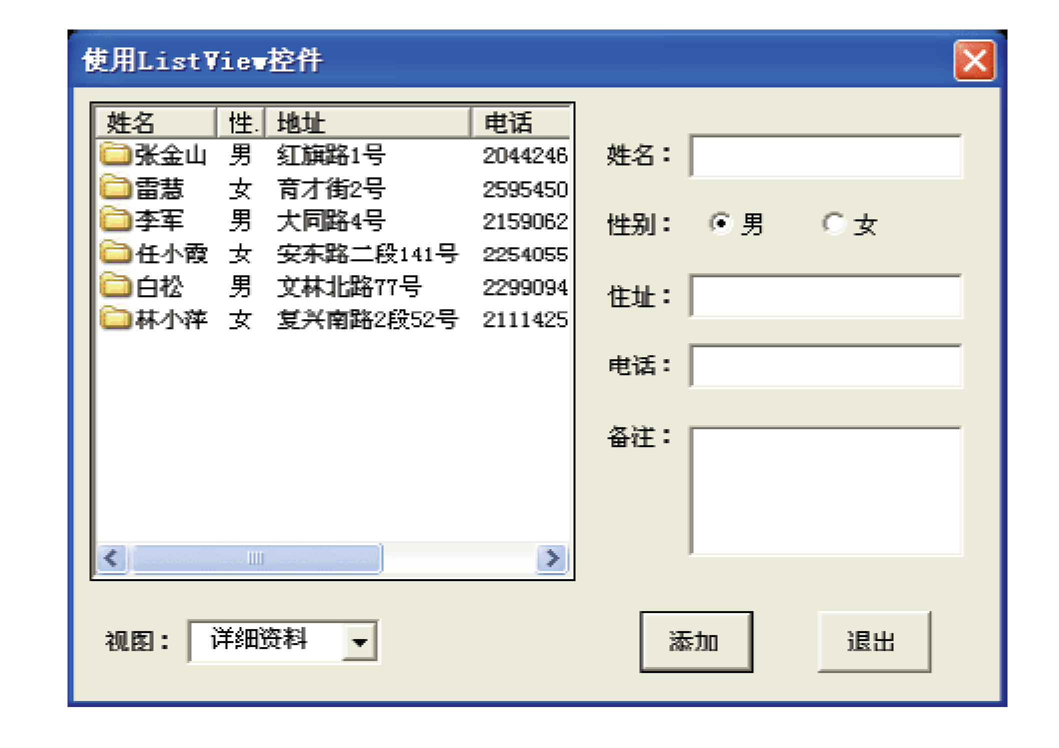


图 19-36　详细资料视图

（14）在用户窗体右侧输入内容，如图 19-37 所示。

（15）输入数据后，单击【添加】按钮即可将输入的内容添加到 ListView 控件中，如图 19-38 所示。



图 19-37　输入数据



图 19-38　添加节点

第 20 章　使用 RibbonX

对用户来说，操作界面是 Excel 2007 中改动最大之处。在 Excel 2007 中，使用 RibbonX（功能区）代替了传统的菜单和工具栏。另外，Excel 2007 采用 Office Open XML 文件格式保存工作簿。用户通过 XML 可直接访问工作簿中的内容，并可通过 XML 自定义 RibbonX。

20.1　了解 Office（2007）Open XML 文件格式

Microsoft Office 2007 引入了一种基于 XML 的新文件格式。这种新格式称为 Microsoft Office Open XML Formats，适用于 Word 2007、Excel 2007 和 PowerPoint 2007。

20.1.1　Office Open XML 的优点

Office Open XML 有许多优点，它不仅适用于开发人员及其构建的解决方案，而且适用于个人以及各种规模的组织。主要有以下优点：

1．压缩文件

Office Open XML 文件会自动压缩，某些情况下最多可缩小 75%。Office Open XML 使用 zip 压缩技术来存储文档，由于这种格式可以减少存储文件所需的磁盘空间，并可以降低通过电子邮件、网络和 Internet 发送文件时所需的带宽，因而可能节省成本。在打开文件时，这种格式可以自动解压缩；而在保存文件时，这种格式又可以重新自动压缩。

2．改进了受损文件的恢复

使用 Office Open XML 格式保存的文档不是二进制格式，而是使用文本格式以一定的 XML 结构以模块形式进行组织，从而使文件中的不同数据组件彼此分隔。这样，即使文件中的某个组件（例如，图表或表格）受到损坏，文件本身还是可以打开。

3．易于检测到包含宏的文档

根据工作簿是否包含宏，在保存工作簿时文件的扩展名也有所不同。无宏的工作簿扩展名为“.xlsx”，包含宏的工作簿扩展名为“.xlsm”。这样，通过工作簿名称即可知道文件是否包含宏代码。

4．更好的隐私保护和更强有力的个人信息控制

可以采用保密方式共享文档，因为使用文档检查器可以轻松地识别和删除个人身份信

息和业务敏感信息，例如，作者姓名、批注、修订和文件路径。

5. 更好的业务数据集成性和互操作性

将 Office Open XML 作为 Office 2007 发布版产品集的数据互操作性框架意味着：文档、工作表、演示文稿和表单都可以采用 XML 文件格式保存，任何人都可免费使用该文件格式并获得该文件格式的许可证，而不必支付版权费。Office 还支持客户定义的 XML 架构，用于增强现有 Office 文档类型的功能。这意味着客户在现有系统中可以轻松解除信息锁定，使用熟悉的 Office 程序对相应信息进行操作。在 Office 中创建的信息很容易由其他业务应用程序所采用。打开和编辑 Office 文件只需要一个 ZIP 实用工具和一个 XML 编辑器即可。

6. 向后兼容性

Office Open XML 是向后兼容的，它可以兼容早期的版本，例如 Office 2000，Office XP 和 Office 2003。这些版本的用户下载一个免费的更新，即可在以前版本中打开 Office 2007 的文档。

20.1.2 Excel 2007 Open XML 文件结构

下面通过对如图 20-1 所示的工作簿文件进行剖析，以使读者深入地了解 Excel 2007 Open XML 文件的结构。

Excel 2007 Open XML 文件其实是一个 zip 文件。为了分析其结构，需要将其解压出来。因此，计算机系统中需要安装管理 zip 文件的软件（例如 WinZip、WinRar 等）。下面列出具体的步骤：

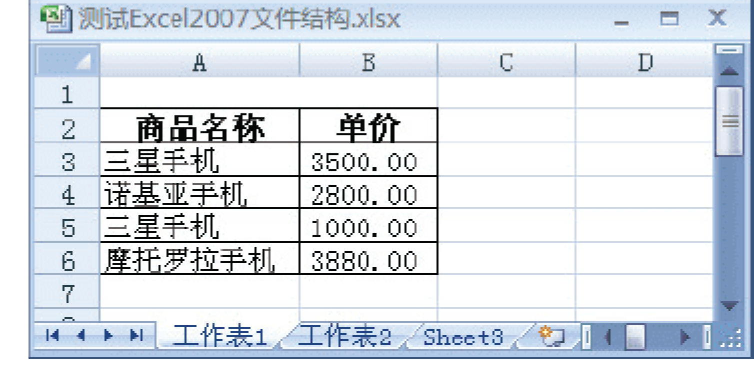


图 20-1 Excel 2007 工作簿

（1）将工作簿文件名“测试 Excel2007 文件结构.xlsx”的后面添加一个“.zip”扩展名，修改为“测试 Excel2007 文件结构.xlsx.zip”。

（2）接着当弹出如图 20-2 所示的警告信息，单击【是】按钮，完成文件名的修改。

（3）将上步更名的文件解压，得到如图 20-3 所示的文件目录结构。

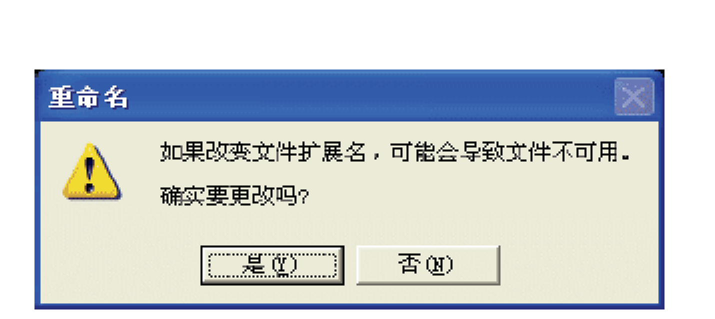


图 20-2 警告信息



图 20-3 解压后的文件结构

（4）在解压后的文件夹中，最重要的文件是文件夹 xl 中的 workbook.xml 文件，该文件的代码如下：

```
  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <workbook xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/
main" xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/
relationships">
  <fileVersion appName="xl" lastEdited="4" lowestEdited="4" rupBuild=
  "4505" />
  <workbookPr defaultThemeVersion="124226" />
- <bookViews>
  <workbookView xWindow="480" yWindow="15" windowWidth="8670" window
  Height="5070" />
  </bookViews>
- <sheets>
  <sheet name="工作表 1" sheetId="1" r:id="rId1" />
  <sheet name="工作表 2" sheetId="2" r:id="rId2" />
  <sheet name="Sheet3" sheetId="3" r:id="rId3" />
  </sheets>
  <calcPr calcId="125725" />
  </workbook>
```

workbook.xml 文件包含一对<sheets>标签，其中的每个<sheet>元素都代表 Excel 2007 文件中的一个，工作表的名称就是其 name 属性的值，在上面的代码中，分别有 3 个工作表，即工作表 1、工作表 2 和工作表 3。<sheet>元素中 r:id 属性的值指出保存工作表数据的 XML 文件。

（5）打开“xl/_rels/workbook.xml.rels”文件，其内容如下所示：

```
  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
-  <Relationships  xmlns="http://schemas.openxmlformats.org/package/2006/
relationships">
  <Relationship Id="rId3"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationsh-
ips/worksheet"
Target="worksheets/sheet3.xml" />
  <Relationship Id="rId2"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationsh-
ips/worksheet"
Target="worksheets/sheet2.xml" />
  <Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationsh-
ips/worksheet"
Target="worksheets/sheet1.xml" />
  <Relationship Id="rId6"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationsh-
ips/sharedStrings"
Target="sharedStrings.xml" />
  <Relationship Id="rId5"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationsh-
ips/styles"
```

```
Target="styles.xml" />
  <Relationship Id="rId4"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationsh-
ips/theme"
Target="theme/theme1.xml" />
  </Relationships>
```

在以上文件中，根据<sheet>元素中 r:id 属性的值可得到工作表数据的 XML 文件。例如，在 workbook.xml 文件中名为工作表 1 的工作表的 r:id 属性为 rld1，在以上文件中根据 ID 找到以下代码：

```
  <Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationsh-
ips/worksheet"
Target="worksheets/sheet1.xml" />
```

由此可知工作表数据保存在 worksheets 文件夹下，文件名为 sheet1.xml。

（6）打开“xl\worksheets\sheet1.xml”文件，其内容如下（为节省篇幅，以下代码中省略了重复的 4～6 行的数据）：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <worksheet xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/
main"
xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relation
ships">
  <dimension ref="A1:B6" />                  '工作表数据范围
- <sheetViews>
- <sheetView tabSelected="1" workbookViewId="0">
  <selection activeCell="D12" sqref="D12" />
  </sheetView>
  </sheetViews>
  <sheetFormatPr defaultRowHeight="13.5" />
- <cols>
  <col min="1" max="1" width="13" bestFit="1" customWidth="1" />
  </cols>
- <sheetData>                               '工作表数据
- <row r="1" spans="1:2">                   '第 1 行数据
  <c r="A1" s="5" />
  <c r="B1" s="5" />
  </row>
- <row r="2" spans="1:2" ht="14.25">        '第 2 行数据
- <c r="A2" s="1" t="s">                     '单元格 A2 的值，字符型
  <v>0</v>  '字符串的位置索引
  </c>
- <c r="B2" s="1" t="s">                     '单元格 B2 的值，字符型
  <v>1</v>
  </c>
  </row>
- <row r="3" spans="1:2">                   ' 第 3 行数据
- <c r="A3" s="2" t="s">
```

```
  <v>2</v>
  </c>
- <c r="B3" s="3">                  '单元格 B3 的值
  <v>3500</v>                       '值为 35000
  </c>
  </row>
      （此处省略工作表第 4~6 行的数据）
  </sheetData>
- <mergeCells count="1">
  <mergeCell ref="A1:B1" />         '合并单元格
  </mergeCells>
  <phoneticPr fontId="1" type="noConversion" />
  <pageMargins left="0.7" right="0.7" top="0.75" bottom="0.75" header="0.3"
  footer="0.3" />
  </worksheet>
```

以上 XML 代码中，元素<c>表示该行中的一个单元格，对于单元格中的值，如果<c>元素有“t”属性的话，<c>元素的子元素<v>的值就是各工作表共享的字符串的索引。否则，<v>元素的值就是该单元格的值。

（7）在工作簿中，各工作表使用的字符串统一存放在“xl/sharedStrings.xml”文件中，该文件的内容如下：

```
  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <sst xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
count="6" uniqueCount="5">
- <si>
  <t>商品名称</t>
  <phoneticPr fontId="2" type="noConversion" />
  </si>
- <si>
  <t>单价</t>
  <phoneticPr fontId="2" type="noConversion" />
  </si>
- <si>
  <t>三星手机</t>
  <phoneticPr fontId="2" type="noConversion" />
  </si>
- <si>
  <t>诺基亚手机</t>
  <phoneticPr fontId="2" type="noConversion" />
  </si>
- <si>
  <t>摩托罗拉手机</t>
  <phoneticPr fontId="2" type="noConversion" />
  </si>
  </sst>
```

每组字符串使用元素<si>表示，其排列顺序就是其序号，表示工作表数据的 XML 文件用该序号来引用字符串。

20.2　RibbonX 控件简介

在 Excel 2007 中，提供了一千七百多个 RibbonX 控件。不能使用 VBA 代码从功能区中添加或删除 RibbonX 控件，只能通过编写 XML 代码来完成定制 RibbonX 的工作，且必须将该 XML 代码包含到工作簿文件中，以达到定制 RibbonX 的目的。

使用 XML 代码可描述 RibbonX，指定控件出现的位置、外观、激活时的动作等。RibbonX 通过回调过程与 VBA 进行接口，可以在 VBA 中编写回调过程，以使 RibbonX 中的控件完成不同的功能。

在本书第 1 章中介绍了 RibbonX 的基本组成，本节将从开发人员的角度介绍 RibbonX 的各种控件。

20.2.1　基本控件

基本控件可以添加到自定义组中或者可以包含在其他容器控件中。常用的基本控件有以下几种。

1．标签

标签控件如图 20-4 所示。与用户窗体中的标签控件类似，RibbonX 中的标签控件只作为文字提示信息使用，不响应用户的动作。

标签控件的 XML 标识为<labelcontrol>。

2．复选框

复选框如图 20-4 所示，可通过单击来切换状态，常用于控制一个 UI 控件是否可见。

复选框控件的 XML 标识为<checkbox>。

3．分隔条

分隔条如图 20-4 所示，用于提供组中控件可见的分隔垂直条。

分隔条控件的 XML 标识为<separator>。



图 20-4　标签、复选框和分隔条控件

4．按钮

按钮如图 20-5 所示，这是最普通的控件。按钮可带有图标和标题，能接收用户的单击，并调用相应的 VBA 过程完成任务。如本章前面的例子中，就为按钮指定了一个 VBA 代码。

按钮控件的 XML 标识为<button>。

5．切换按钮

切换按钮如图 20-5 所示，是一个可单击项，每次单击时在按下和非按下之间切换，常

用于组中以切换多个可能状态其中之一的属性，该组中每次仅一个按钮能被按下。

切换按钮控件的 XML 标识为<toggleButton>。

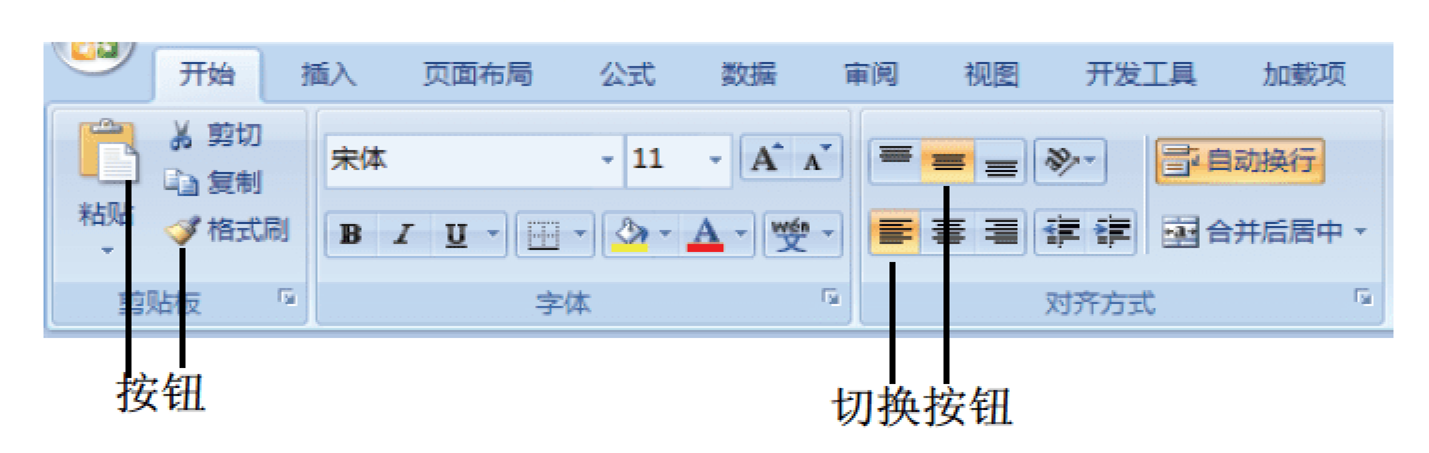


图 20-5　按钮和切换按钮

6．编辑框

编辑框如图 20-6 所示，可接受用户的输入。

编辑框控件的 XML 标识为<editBox>。

7．下拉库列表

下拉库列表如图 20-7 所示，是由一个下拉控件和一组其他控件组成的。库列表中可以包含不同类型的控件，是最灵活的 UI 控件之一。

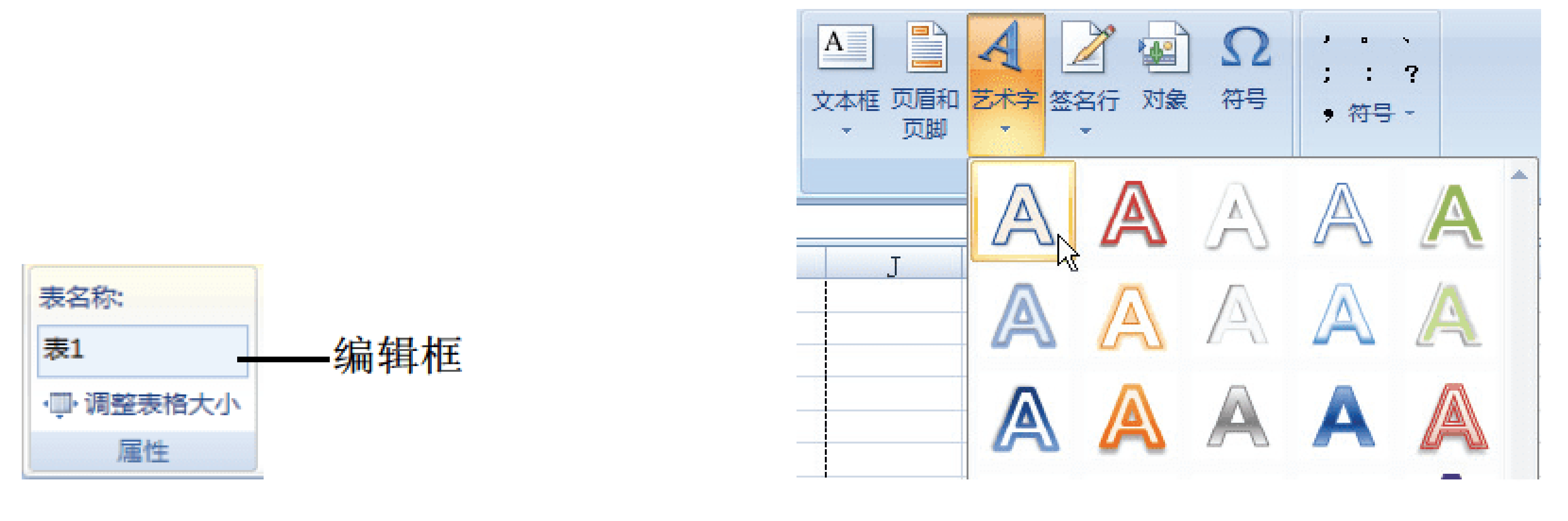


图 20-6　编辑框　　　　图 20-7　下拉库列表

下拉库列表控件的 XML 标识为<gallery>。

8．列表项

为包含在组合框中的列表项。

列表项控件的 XML 标识为<item>。

20.2.2　容器控件

容器控件可以包含其他控件，通过嵌套容器控件在其他容器里，可以创建层次结构。常用的容器控件有以下几种。

1．选项卡

在 Excel 2007 中，功能区被分为多个选项卡，例如【开始】、【插入】选项卡等。

选项卡的 XML 标识为<tab>。

2．组

每个选项卡中又分多个组，例如【开始】选项卡中的【字体】、【对齐方式】等组，各组中包含不同的控件。

组的 XML 标识为<group>。

3．盒子控件

在功能区中，盒子控件没有任何外观，也就是说盒子控件不显示出来，主要用来控制其他按钮的布局。在盒子控件中可以包含任何其他类型的控件。如图 20-8 所示就是使用 box 控件将排序的三个按钮组合在一起。

盒子控件的 XML 标识为<box>。可将其他任意控件放在该容器内。

4．按钮组控件

按钮组控件用来控制其他控件的布局，在相关控件之间带有边框和分隔条，如图 20-9 所示。



图 20-8　盒子控件

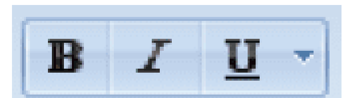


图 20-9　按钮组控件

按钮组控件的 XML 标识符为<buttonGroup>。可将<button>、<toggleButton>、<gallery>、<menu>、<splitButton>等控件放在该容器中。

5．复合框控件

复合框控件如图 20-10 所示，可以在该控件中输入内容，同时提供下拉列表供用户直接选择。下拉列表的内容是一组列表项目（为一组<item>控件）。

复合框控件的 XML 标识为<comboBox>。

6．菜单控件

功能区中的菜单控件是一种弹出式菜单，如图 20-11 所示。其组成元素由 RibbonX 文件定义。菜单可以包含按钮或其他菜单项，且允许创建层次菜单结构。



图 20-10　复合框控件

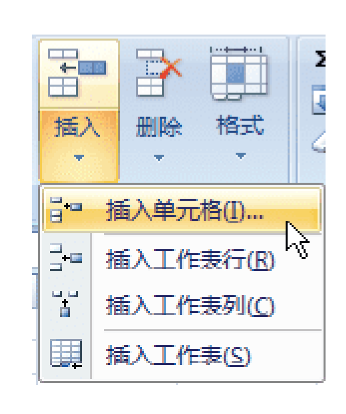


图 20-11　菜单控件

菜单控件的 XML 标识为<menu>。在该容器中可以包含<button>、<toggleButton>、<checkbox>、<gallery>、<Menu>、<splitButton>等控件。

7．分离按钮控件

分离按钮控件由一个组合按钮和菜单或者切换按钮和菜单组成，如图 20-12 所示。单击按钮部分将执行默认的动作，而单击下拉箭头将显示相关选项并可进行选择。

分离按钮控件的 XML 标识为<splitButton>。该容器可以包含<button>、<menu>、<toggleButton>等控件。定义该控件的代码结构如下：

```
<splitButton...>
  <button.../>
  <menu...>
    菜单内容
  </menu>
</splitButton>
```

8．下拉控件

下拉控件如图 20-13 所示，该控件与复合框控件类似。不同的是，在该容器中还可以包含<button>控件。

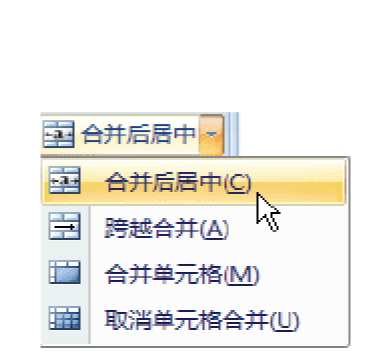


图 20-12　分离按钮控件

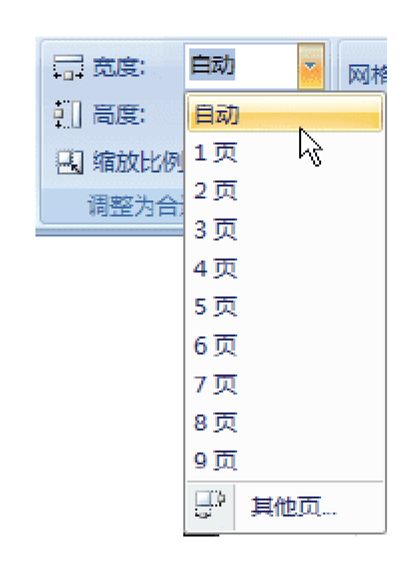


图 20-13　下拉控件

下拉控件的 XML 标识为<dropDown>。在该容器中可包含<item>和<button>控件。

20.2.3　控件属性

每个控件都具有很多的属性，可在 XML 中使用这些属性修改控件的外观。下面列出控件的常用属性。

1．ID类属性

每个控件都必须有一个唯一的 ID，由一个最多是 1024 个字符的字符串组成，ID 属性分为以下 3 类：

- id：为自定义控件的 ID；
- idMso：为内置控件的 ID；

- idQ：为标记<tab>、<group>和<menu>的 ID。

2．外观属性

外观属性用来控制控件的可见性、标签、悬浮提示等信息，常用的有以下属性。

- itemHeight：设置 gallery 控件中库项目的高度，以像素为单位；
- itemWidth：设置 gallery 控件中库项目的宽度，以像素为单位；
- itemSize：设置 menu 中项目的尺寸；可设置为 normal 和 large，大项目不但显示描述也显示标签；
- label：设置控件的标题；
- screentip：设置鼠标悬浮在控件上时显示的小提示；
- supertip：设置鼠标悬浮在控件上时显示的大提示；
- showLabel：设置是否显示控件标签，取值为 True 和 false；
- size：设置控件的尺寸。可设置为 normal 和 large，控件正常尺寸（normal）占用一行，大尺寸（large）占用三行；
- title：设置菜单的标题文本，用于 menu 和 menuSeparator 控件；
- visible：设置控件是否可见，取值为 True 和 false。

3．图像属性

带图像的控件可以使用自定义的图像，也可使用内置控件的图像。主要有以下属性：

- image：使用此属性设置自定义图像的名称；
- imageMso：使用此属性设置内置控件的名称，用来引用内置控件的图像；
- showImage：设置是否显示一个控件的图像，取值为 True 和 false。

4．其他属性

除了上面介绍的控制控件外观、图像的属性外，常用的属性还有以下几种：

- boxStyle：设置在 box 控件中是水平排列图标（默认）或垂直排列图标，取值为 Horizontal 或 vertical；
- columns：设置 gallery 控件库中的列数；
- description：设置控件的长描述，当菜单的 itemSize 设置为大时显示在菜单中，用于 botton、toggleButton、splitButton、checkBox、menu、dynamicMenu、gallery 等控件；
- enabled：设置控件是否有效，如果设为 false，则该控件为灰色，不能操作；
- maxLength：设置 editBox 或 comboBox 控件文字输入的最大长度；
- keytip：设置访问控件的快捷键；
- rows：设置 gallery 控件库中的行数；
- showItemImage：设置在 comboBox、dropDown 或 gallery 控件中，是否在下拉列表中显示图像；
- tag：为控件设置附加文本。

20.2.4　控件回调函数

在自定义 RibbonX 时，可在 XML 中通过控件的不同属性设置控件。但是，在更多的情况下，需要程序在运行时修改控件的属性。这时，可使用控件的回调函数。

在 RibbonX 中提供了回调函数功能，可在运行时动态修改控件的属性。每个控件可有多个回调函数，可分别动态修改控件的相应属性。设置回调函数属性以 get 字符开头，紧跟着相应的属性名。例如，要动态修改控件的 label 属性，则用属性 getLabel 设置回调函数。使用属性 getItemImage 设置的回调函数可在程序运行时修改控件的图像。

还有 3 个不是按这种方式命名的属性，也可用来设置回调函数。

- onLoad：用在 XML 文件的最外层元素<customUI>中，当 Office 开始装载 RibbonX 时将调用该属性指定的回调函数。
- onAction：当单击控件时，调用该属性指定的 VBA 过程，可应用到<button>、<toggleButton>、<checkBox>、<dropdown>和<gallery>控件。
- onChange：当控件中的文本改变时，调用该属性指定的 VBA 过程，可应用到<editBox>和<comboBox>控件。

例如，以下 XML 代码定义一个<button>控件，并指定单击按钮时调用名为 showmsg 的 VBA 过程。

```
<button id="b1"  imageMso=" HappyFace"  size="large"
   label="测试"  onAction="showmsg"/>
```

在 Excel 工作簿中，编写一个名为 showmsg 的过程，执行相应的功能即可。

```
Sub showmsg(control As IRibbonControl)
   MsgBox "您单击了“测试”按钮!"
End Sub
```

从上面的代码可以看出，在 VBE 中编写回调函数时，过程定义部分需要指定参数。不同的回调函数的参数数量不同，下面介绍定义回调函数子过程的结构。

1．onLoad属性设置回调函数

onLoad 属性只能在 XML 文件的最外层元素<customUI>中使用，该属性设置系统装载自定义 RibbonX 时执行的回调函数名称，在程序中只能通过该回调函数获取 RibbonX 对象的引用。onLoad 属性指定的回调函数的过程结构如下：

```
Sub 过程名(ribbon As IRibbonUI)
```

2．onAction属性设置回调函数

在 XML 中用控件的 onAction 属性指定回调函数，针对不同控件有不同的子过程结构。

- 控件为<button>时，onAction 属性指定的回调函数的过程结构如下：

```
Sub 过程名(ByRef  Control  As  IRibbonControl)
```

- 控件为<checkBox>和<toggleButton>时，onAction 属性指定的回调函数的过程结构如下：

```
Sub 过程名(ByRef  Control  As  IRibbonControl, ByRef  Pressed  As  Boolean)
```

- 控件为<dropDown>和<gallery>时，onAction 属性指定的回调函数的过程结构如下：

```
Sub  过 程 名 (ByRef   Control   As   IRibbonControl,ByRef   SelectedID   As
String,_
  ByRef  SelectedIndex  As  Integer)
```

3. onChange属性设置回调函数

控件为<editBox>和<comboBox>时，onChange 属性指定的回调函数的过程结构如下：

```
Sub 过程名(ByRef  Control  As  IRibbonControl,ByRef  Text  As  String)
```

4. 其他get类属性设置回调函数

- 通过属性 getItemID、getItemImage、getItemLabel、getItemScreentip、getItemSupertip 指定的回调函数的过程结构如下：

```
Sub 过程名(ByRef  Control  As  IRibbonControl,ByRef  Index  As  Integer, _
  ByRefReturnValue As Variant)
```

- 除前面列出各属性设置的回调函数外，其他 get 属性（包括 getContent、getDescription、getEnabled、getImage、getItemCount、getItemHeight、getItemWidth、getKeytip、getLabel、getPressed、getSize、getScreentip、getSelectedItemID、getSelectedItemIndex、getShowImage、getShowLabel、getSupertip、getText、getTitle、getVisible 等属性）指定的回调函数的过程结构如下：

```
Sub  过 程 名 (ByRef   Control   As   IRibbonControl,  ByRef   ReturnValue   As
Variant)
```

20.3　自定义 RibbonX

在 20.2 节中介绍了 RibbonX 的常用控件，以及控件的属性、回调函数。了解这些内容之后，就可使用 XML 自定义 RibbonX。最简单的方式就是手工方式：使用【记事本】编写 XML 代码，再添加到 Excel 工作簿中。另外，还可使用 Custom UI Editor 工具快速地定义 RibbonX。

20.3.1　手工方式自定义 RibbonX

本章的 20.1 节中介绍了 Excel 2007 工作簿的 Oopen XML 格式。要自定义 RibbonX，也需要将工作簿解压出来，并向其中添加自定义代码。下面的例子向功能区添加一个名为

【测试】的选项卡，在该选项卡中添加一个按钮用来显示工作表的信息，如图 20-14 所示。

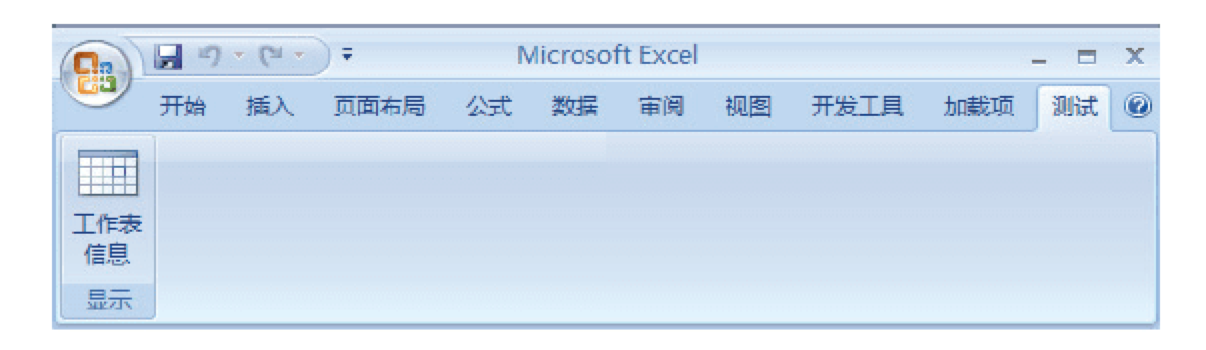


图 20-14　自定义 RibbonX

按以下步骤完成上面的自定义选项卡：

（1）在当前文件夹中创建一个名为 customUI 的文件夹。

（2）打开【记事本】程序，输入以下内容：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab id="rxtabTest" label="测试" >
          <group id="myGroup" label="显示">
              <button id="b1" imageMso="AccessTableEvents"
                    size="large"  label="工作表信息" onAction="showmsg"/>
          </group>
     </tab>
   </tabs>
  </ribbon>
</customUI>
```

注意：因为 XML 要区分大小写，所以一定要注意字母的大小写。

以上代码使用 XML 自定义 RibbonX。有关 XML 的内容请读者参阅 XML 相关书籍。下面简单介绍本例中用到的元素。

- <customUI>元素是 XML 的根容器，名称集（namespace）将它识别作为 RibbonX 文档。
- <ribbon>元素是一个联系到可见的 Ribbon 的所有变化的容器。<customUI>元素也可以包含一个<commands>元素，用来重复利用内置控件。
- <tabs>元素是一个联系到 Ribbon 中现有的或新的选项卡的所有变化的容器。<ribbon>元素也能包含<officeMenu>、<qat>和/或<contextualTabs>元素来控制 Ribbon 的相应部分。
- < tab id="rxtabTest" label="测试">元素创建自定义的选项卡。
- < group id="myGroup" label="显示">元素创建一个组。
- <button>元素添加一个按钮，该按钮显示名称为“工作表信息”，当单击该按钮时执行工作簿中的 showmsg 宏，该宏需要在 Excel 的 VBE 中编写。
- 接下来就使用</group>、</tab>等代码结束各元素的定义。

（3）选择【文件】|【保存】命令，打开如图 20-15 所示的【另存为】对话框。在【保存类型】下拉列表框中选择“所有文件”，在【编码】下拉列表框中选择 UTF-8，将文件

保存到当前文件夹的 customUI 文件夹下，名称为 customUI.xml。

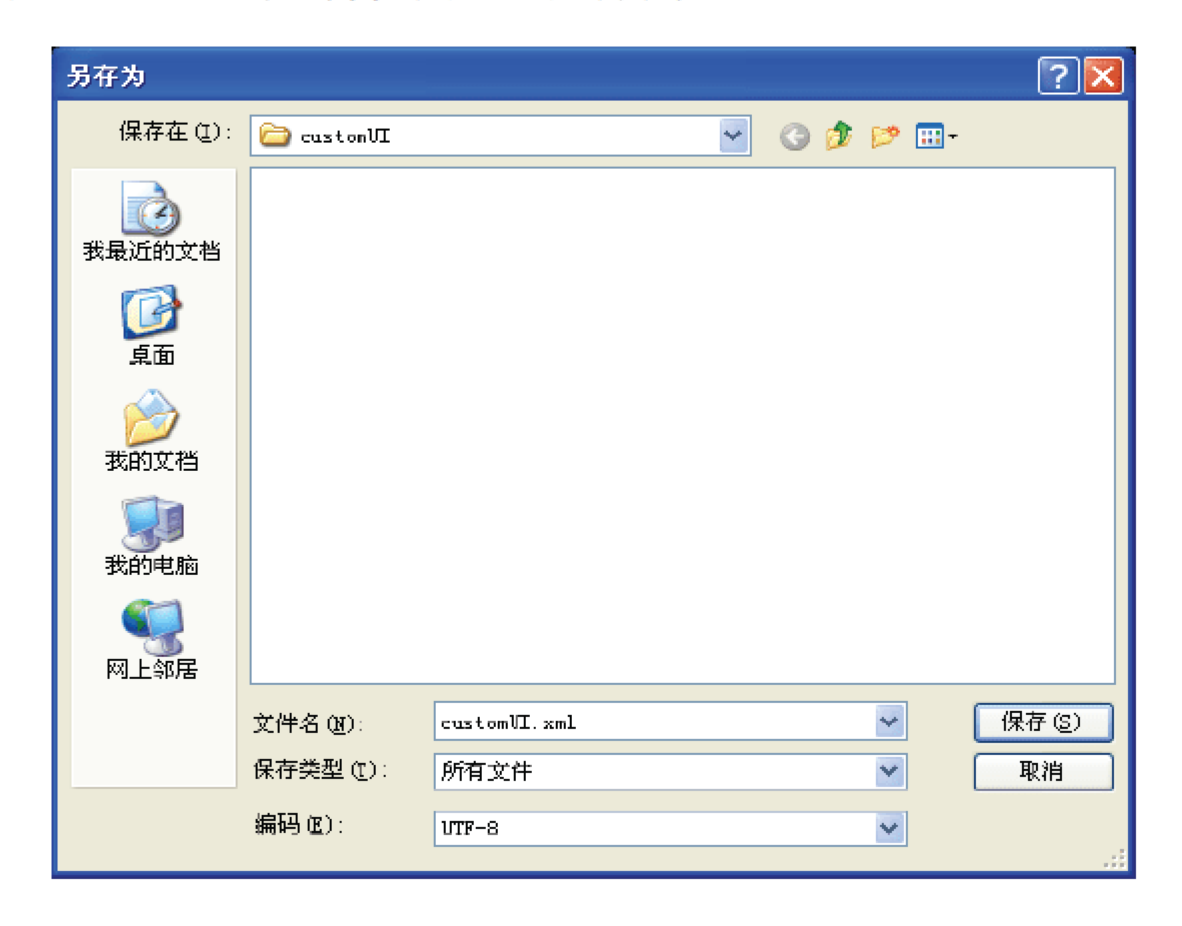


图 20-15　保存 XML 文件

（4）打开 Excel 2007，新建一个工作簿，保存为 Test.xlsm。

（5）关闭 Excel 2007。并将文件 test.xlsm 重命名为 test.xlsm.zip，使 Excel 工作簿变为一个压缩文件。

（6）双击压缩文件用 WinRar 打开该文件，如图 20-16 左图所示。拖动当前文件夹下的 customUI 文件夹到打开的压缩文件窗口，得到如图 20-16 右图所示效果，将该文件夹及文件夹的内容添加到压缩文件中。



图 20-16　添加文件到压缩文件

（7）将图 20-16 所示对话框中的_rels 文件夹拖到当前文件夹中。

（8）打开_rels 文件夹中的.rels 文件，在最后一个 Relationship 标记与 Relationships 标记之间添加以下内容，将在工作簿文件与 customUI 文件夹中的 customUI.xml 文件之间创建关系。

```
<Relationship Id="customUIRelID"
Type="http://schemas.microsoft.com/office/2006/relationships/ui/extensi
bility" Target="customUI/customUI.xml" />
```

（9）保存并关闭.rels 文件。

（10）删除图 20-16 所示对话框中的_rels 文件夹，然后拖动第（7）步修改_rels 文件夹到压缩文件 Test.zip 中。

（11）关闭压缩文件窗口，将压缩文件 Test.xlsm.zip 重命名为 Test.xlsm。

（12）用 Excel 2007 打开 Test.xlsm 工作簿，将看到如图 20-14 所示的自定义选项卡。

（13）单击自定义的【测试】选项卡中的【工作表信息】按钮，因为还未编写回调函数，所以将弹出如图 20-17 所示的错误提示对话框。

（14）按组合键 Alt+F11 打开 VBE 环境。

（15）单击主菜单【插入】|【模块】命令增加一个标准模块。

（16）在【模块 1】中输入以下过程代码：

```
Sub showmsg(control As IRibbonControl)
    Dim str1 As String
    With ActiveSheet
        str1 = "当前工作表信息: " & vbNewLine
        str1 = str1 & "工作表名: " & .Name & vbNewLine
        str1 = str1 & "行: " & .Cells.Rows.Count & vbNewLine
        str1 = str1 & "列: " & .Cells.Columns.Count & vbNewLine
        str1 = str1 & "已使用区域: " & .UsedRange.Address
    End With
    MsgBox str1, vbInformation + vbOKOnly, "提示信息"
End Sub
```

以上代码的作用是：获取活动工作簿的名称、工作表行列数、已使用区域的地址，并使用 MsgBox 对话框显示出来。

（17）返回 Excel 界面，单击自定义的【测试】选项卡中的【工作表信息】按钮，将弹出如图 20-18 所示的提示对话框，显示当前工作表的信息。

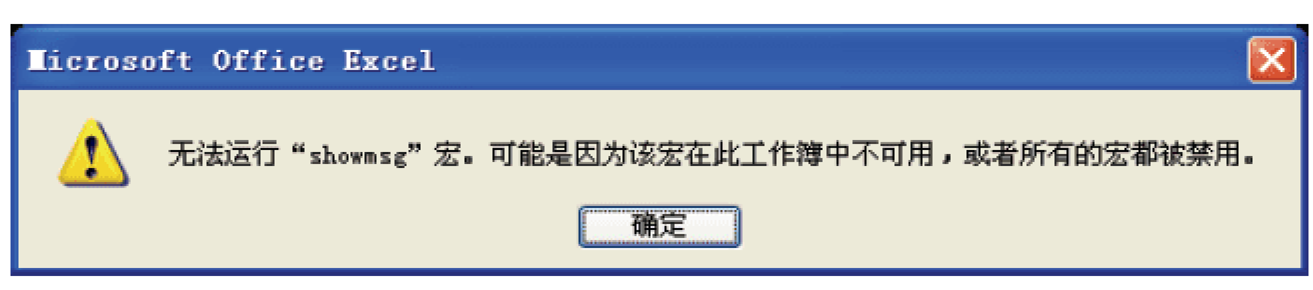




图 20-17　错误提示

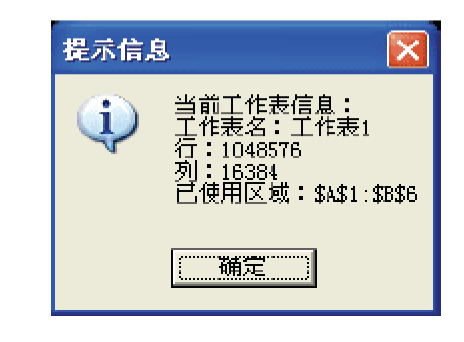


图 20-18　提示对话框

20.3.2　使用 UI 编辑器自定义 RibbonX

使用手工方式自定义 RibbonX 时，操作步骤比较繁琐。首先，需要将工作簿文件修改为压缩文件扩展名，将自定义 RibbonX 的 XML 代码添加到压缩文件中，其次，需要修改压缩文件中_rels 文件夹中的文件，最后再将压缩文件名称改回 Excel 工作簿名称。

为了快速、高速地完成 RibbonX 的定义，微软公司提供了一个名为 Custom UI Editor 的工具，可用来处理存放在 Excel Open XML 工作簿中自定义 RibbonX 的实用程序。使用该工具软件能够自动处理放置自定义 UI 部分到压缩中的过程，并定义与它的关系，所有

的处理只需通过单击鼠标完成。

1．认识UI编辑器

读者可从 http://openxmldeveloper.org/articles/customuieditor.aspx 网站上下载 Custom UI Editor 工具软件。

在安装 Custom UI Editor 工具之前，应先检查系统是否安装.NET Framework 2.0。如果没有安装，需要首先下载安装后，再安装 Custom UI Editor 工具。

安装完成后，UI 编辑器将在桌面创建一个快捷方式，双击运行 UI 编辑器，打开如图 20-19 所示的操作界面。

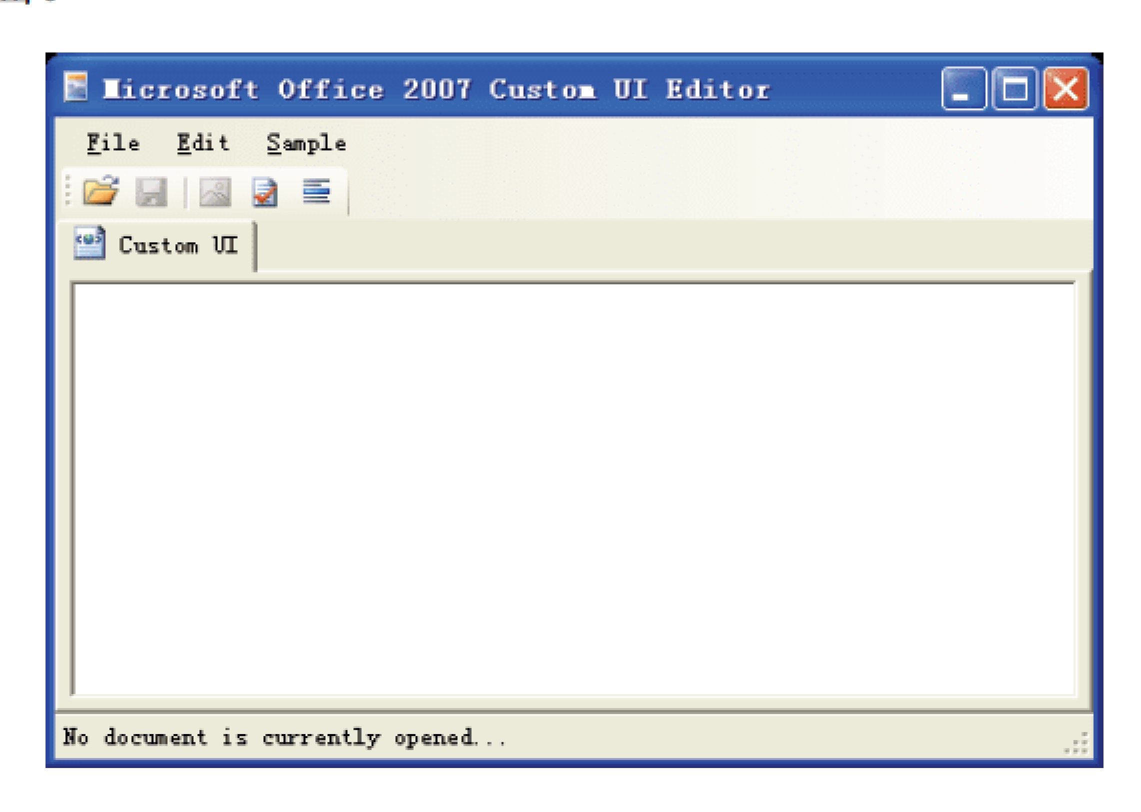


图 20-19　UI 编辑器

UI 编辑器由菜单栏、工具栏和一个文字编辑窗口组成。在文字编辑窗口中可输入自定义 UI 的 XML 代码。使用 UI 编辑器的一般步骤是：

（1）打开一个 Office 2007 格式的工作簿文件。

（2）在 UI 编辑器的编辑窗口中输入自定义 RibbonX 的 XML 代码。

（3）保存文件，退出 UI 编辑器。

（4）打开文件即可查看到自定义 RibbonX 的效果。

从以上步骤可以看出，使用 UI 编辑器可减少将文件改名为 zip、将 customUI 文件加入压缩包、修改.rels 文件等过程，可快速完成自定义 UI 的过程。

2．使用UI编辑器

使用 UI 编辑器自定义 UI 的过程非常简单，下面就使用 UI 编辑器来制作上节的实例。

（1）在 Excel 2007 中新建一个工作簿，保存为 Test2.xlsm，退出 Excel 2007。

（2）打开 UI 编辑器（Custom UI Editor）。

（3）单击菜单 File|Open 命令，在打开的对话框中选择 test2.xlsm 文件，如图 20-20 所示。

（4）单击【打开】按钮，在 UI 编辑器中打开工作簿 test2.xlsm。此时，编辑器窗口仍然显示为空白，只是标题栏显示了工作簿的文件名，如图 20-21 所示。

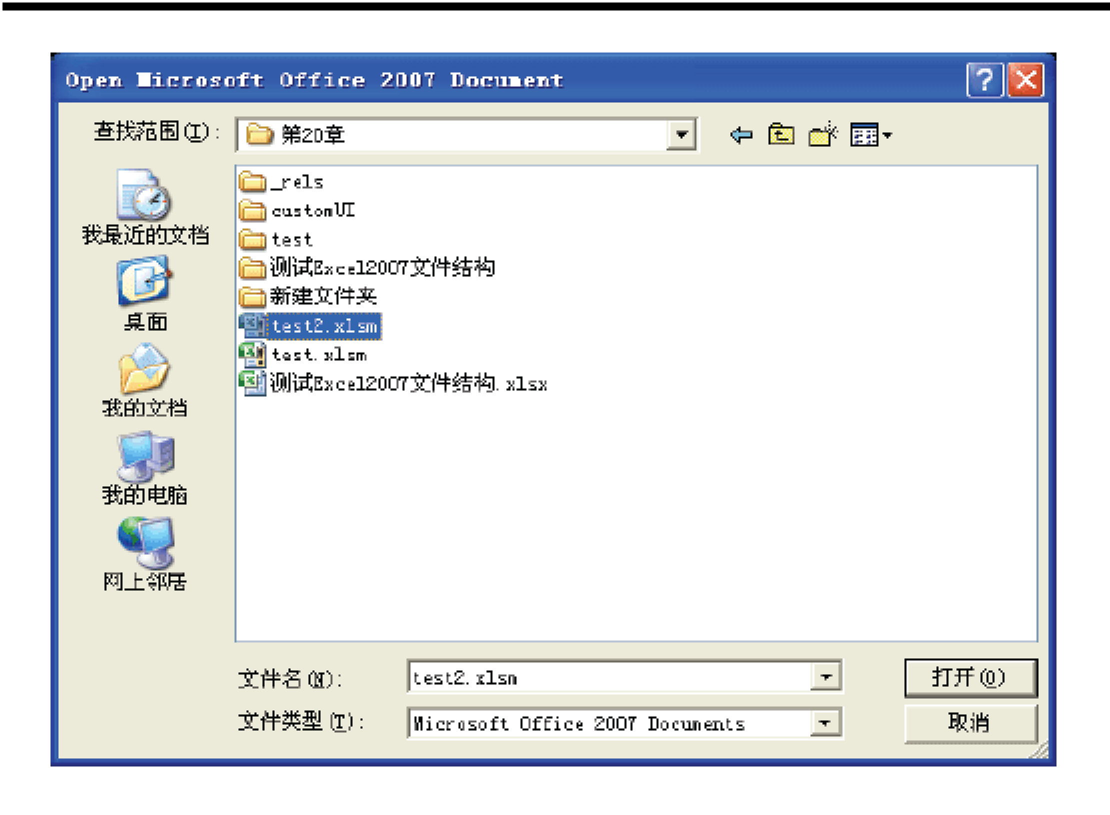


图 20-20　打开文件

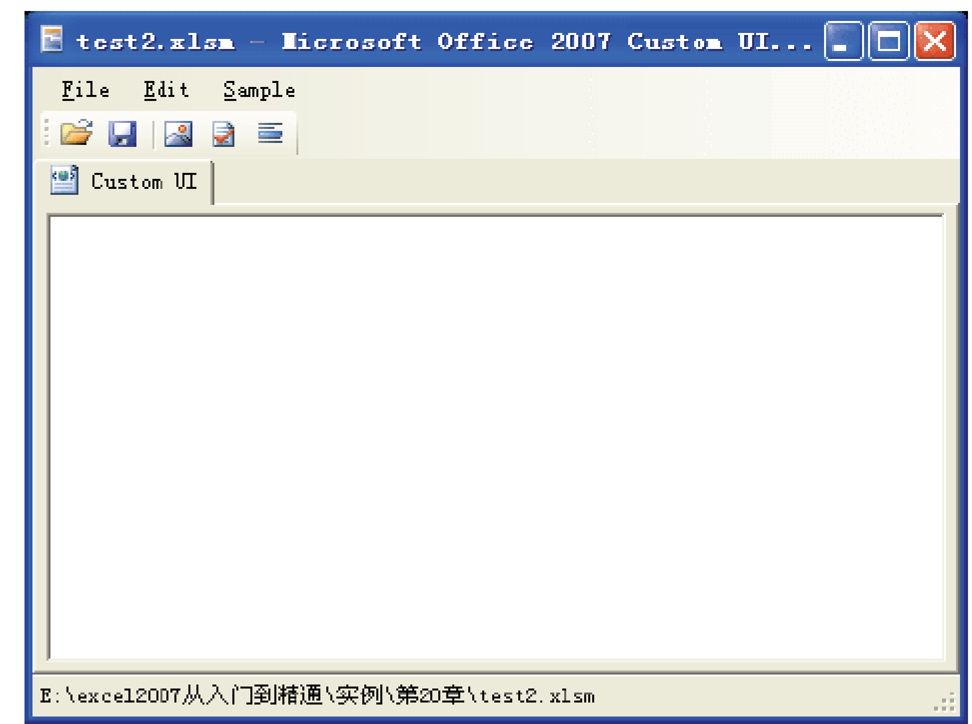


图 20-21　打开工作簿

（5）因为工作簿 Test2.xlsm 中还没有自定义 RibbonX 的代码，所以 UI 编辑器的编辑窗口里为空。在编辑窗口中输入以下 XML 代码，用来定义 RibbonX，如图 20-22 所示。

提示： UI 编辑器不支持中文，因此需要将上节自定义 RibbonX 代码的中文改成英文。

图 20-22　输入 XML 代码

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab id="rxtabTest" label="Test" >
          <group id="myGroup" label="Display">
              <button id="b1" imageMso="AccessTableEvents"
                    size="large" label="Worksheet Info" onAction="showmsg"/>
           </group>
       </tab>
     </tabs>
  </ribbon>
</customUI>
```

（6）单击工具栏中的 Save 按钮将内容保存到 test2.xlsm 文件中，然后关闭 UI 编辑器

完成 RibbonX 的创建。

（7）打开工作簿 test2.xlsm，可看到功能区新增加的选项卡如图 20-23 所示。

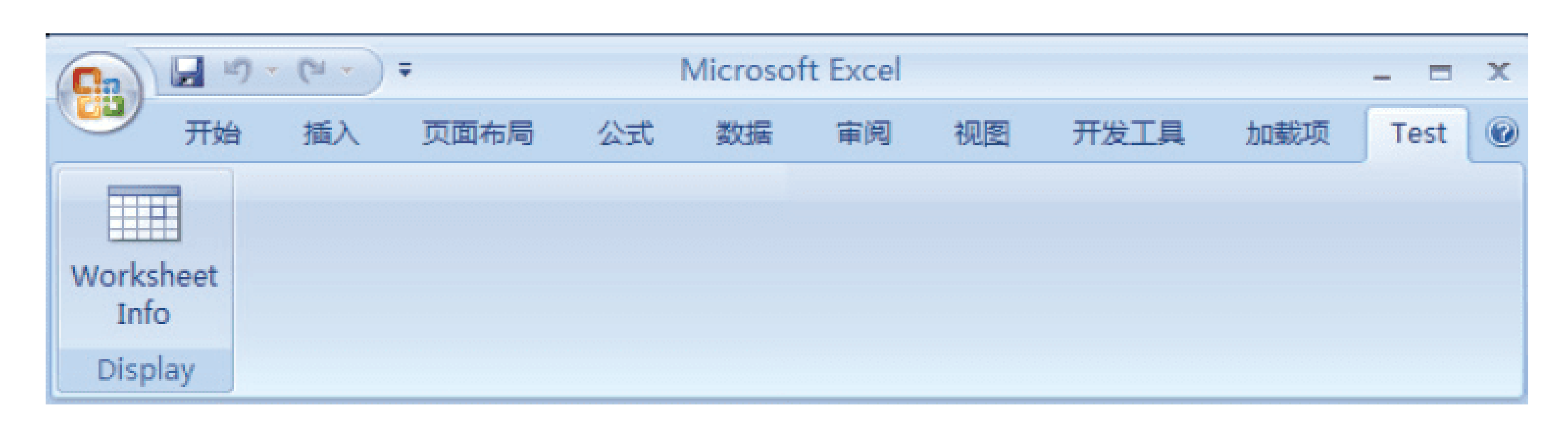


图 20-23　自定义功能区

（8）在 VBE 中，编写过程 showmsg 用来显示当前工作表的信息。具体代码与 20.3.1 节中的过程 showmsg 完全相同。

20.4　自定义 RibbonX 实例

前面介绍了自定义 RibbonX 的步骤、常用 RibbonX 控件等内容。本节以实例形式演示自定义 RibbonX 的各种方法，包括将内置 RibbonX 控件组合成一个新的选项卡、添加自定义 RibbonX 控件到内置选项卡、自定义 Office 按钮功能、定义回调函数等实例。

本节的实例不再重复创建的各个步骤，具体操作可参见本章 20.3 节的内容，在实例中只列出定义 RibbonX 的 XML 代码、回调函数的 VBA 代码等内容。

20.4.1　组合内置 Ribbon

在 Excel 2007 功能区的内置控件中，将不同用途的控件用选项卡进行了分类。在大多数情况下，用户要完成工作都需要在多个选项卡之间来回切换。利用功能区的自定义功能，可以新建一个选项卡，将常用内置控件集中到一起，以方便用户操作。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab id="customTab" label="常用工具" insertAfterMso="TabHome">
        <group idMso="GroupClipboard" />
        <group idMso="GroupFont" />
        <group idMso="GroupSortFilter" />
        <group idMso="GroupInsertIllustrations" />
        <group id="NewGroup" label="组合按钮">
          <button idMso="VisualBasic" />
          <button idMso="SheetProtect" />
          <button idMso="FilePrint" />
          <separator id="MySeparator1" />
          <menu idMso="WindowSwitchWindowsMenuExcel" size="large" />
```

```
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

以上 XML 代码自定义的选项卡如图 20-24 所示。



图 20-24　组合内置控件

对于内置控件，用户不需要编写任何 VBA 代码，单击这些控件即可调用系统提供的功能。在 Excel 2007 中提供了一千七百多个内置 RibbonX 控件，用户要使用这些内置控件必须先要知道具体的名称。可以在 http://www.microsoft.com/zh/cn/default.aspx 网站上以关键字 2007OfficeControlIDsExcel2007 搜索，找到 2007OfficeControlIDsExcel2007.EXE 文件，将其下载到本地硬盘。该文件为一个自解压文件，将其解压缩后可看到其中包含 24 个文件，分别为 Office 2007 各组件（包括 Excel、Word、Outlook 和 PowerPoint）中 RibbonX 控件的名称、类型等。Excel 2007 中 RibbonX 控件名称包含在 ExcelRibbonControls.xlsx 文件中，打开该文件，可看到如图 20-25 所示的控件信息。

ExcelRibbonControls.xlsx

	A	B	C	D
1				
2	Control Name	Control Typ	Tab Set Name	Tab Name
3	FileNewDefault	button	None (Quick Access Toolbar	Quick Access Too
4	FileOpen	button	None (Quick Access Toolbar	Quick Access Too
5	FileSave	button	None (Quick Access Toolbar	Quick Access Too
6	FileSendAsAttachment	button	None (Quick Access Toolbar	Quick Access Too
7	FilePrintQuick	button	None (Quick Access Toolbar	Quick Access Too
8	FilePrintPreview	button	None (Quick Access Toolbar	Quick Access Too
9	Spelling	button	None (Quick Access Toolbar	Quick Access Too
10	Undo	gallery	None (Quick Access Toolbar	Quick Access Too
11	Redo	gallery	None (Quick Access Toolbar	Quick Access Too
12	SortAscendingExcel	button	None (Quick Access Toolbar	Quick Access Too
13	SortDescendingExcel	button	None (Quick Access Toolbar	Quick Access Too
14	FileNew	button	None (Office Menu)	Office Menu
15	FileOpen	button	None (Office Menu)	Office Menu

ExcelRibbonControls

图 20-25　Excel 2007 控件列表

另外，Microsoft Office 2007 提供了大约二千五百个内置命名的图像，这些图像与不同的命令相关联。如果知道图像的名称，则可以为自定义的 RibbonX 控件中指定这些图像。

微软提供的 mso image browser.xlsm 工作簿列出了 Excel 2007 提供的这些图像，可从网上下载该工作簿。该工作簿的内容如图 20-26 所示，在工作表第 1 行中列出了当前选中名称的图像及后面的 50 个图像。从第 2 行开始，每行第 1 列显示 1 个内置图像的名称，在 XML 代码中使用 imageMso 属性引用图像名称，即可显示出对应的图像。

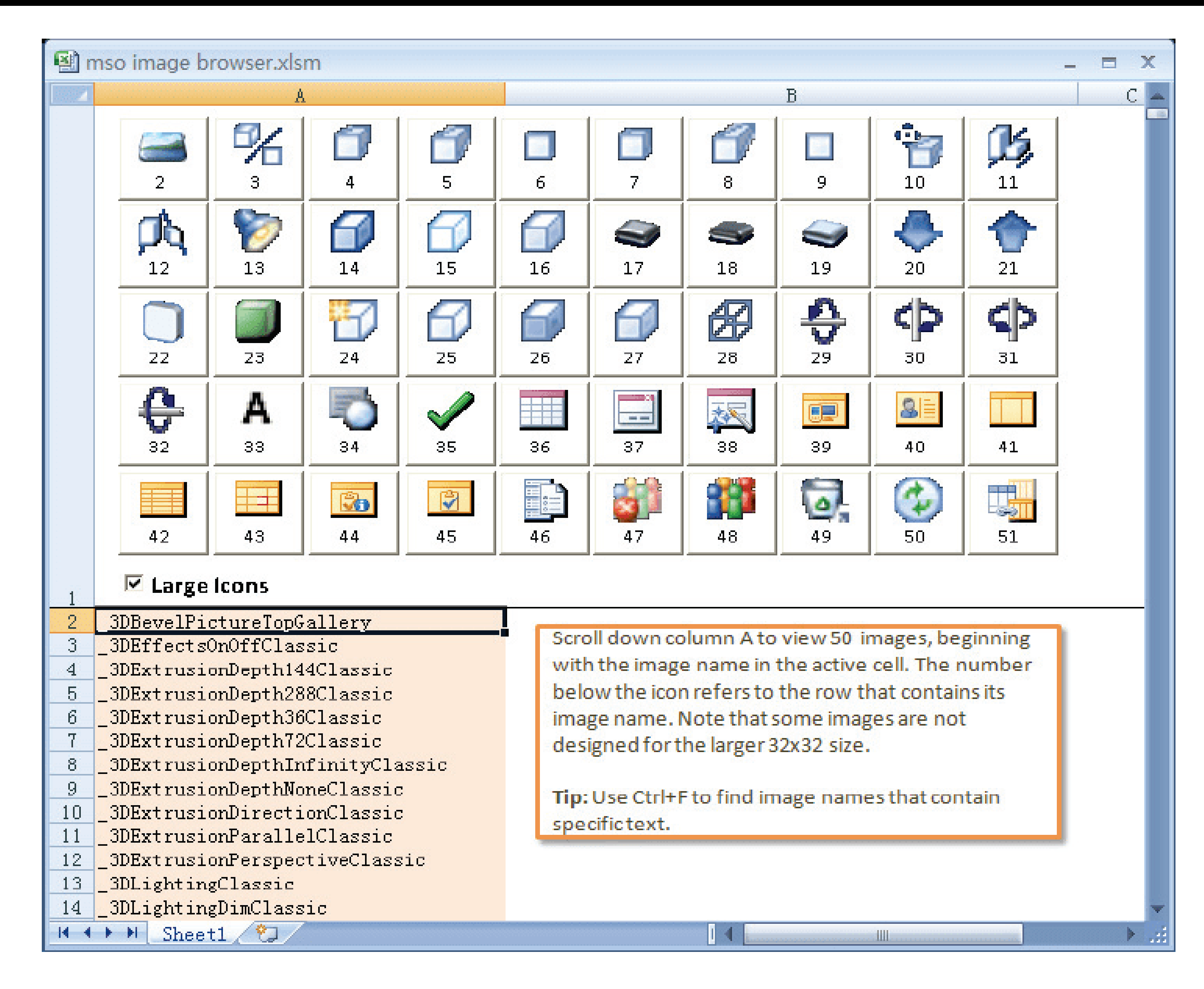


图 20-26　Excel 2007 内置图像

提示：也可以将内置图像显示在用户窗体的 Image 控件中。例如，下面的代码将为用户窗体中的 Image1 控件设置内置图像，图像的尺寸被指定为 32 × 32 像素。

```
Image1.Picture = Application.CommandBars.GetImageMso("AccessTableEvents",
32, 32)
```

20.4.2　添加 RibbonX 到内置选项卡

可将自定义 RibbonX 控件添加到内置选项卡中。例如，以下代码在【开始】选项卡中添加一个名为【测试】的组：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab idMso="TabHome" >
        <group id="Group1" label="测试"  >
          <button id="Button1" label="标签 1" size="normal" onAction="sub1"
              imageMso="DrawingCanvasScale" />
          <button id="Button2" label="标签 2" size="normal" onAction="sub2"
              imageMso="DrawingObjectFormatDialog" />
          <button id="Button3" label="标签 3" size="normal" onAction="sub3"
              imageMso="DrawingCanvasExpand" />

          <separator id="Separator1" />
```

```
          <menu id="Menu" label="测试菜单" size="large"
               imageMso="FileBackUpSqlDatabase"  >
            <button id="Button4" label="菜单项 1"  onAction="MenuSub1"
               imageMso="FileCheckIn" />
            <button id="Button5" label="菜单项 2"  onAction="MenuSub2"
                imageMso="FileCheckOut" />
            <button id="Button6" label="菜单项 3"  onAction="MenuSub3"
               imageMso="FileCheckOutDiscard" />
            <button id="Button7" label="菜单项 4"  onAction="MenuSub4"
               imageMso="FileCompatibilityChecker" />
            <button id="Button8" label="菜单项 5"  onAction="MenuSub5"
                imageMso="FileCompatibilityCheckerPowerPoint" />
            <button id="Button9" label="菜单项 6"  onAction="MenuSub6"
               imageMso="FileCompatibilityCheckerWord" />
          </menu>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

完成设置后打开工作簿，可看到如图 20-27 右侧所示的自定义组“测试”，其中包含了 3 个按钮和 1 个菜单，单击菜单将弹出 6 个菜单项目。

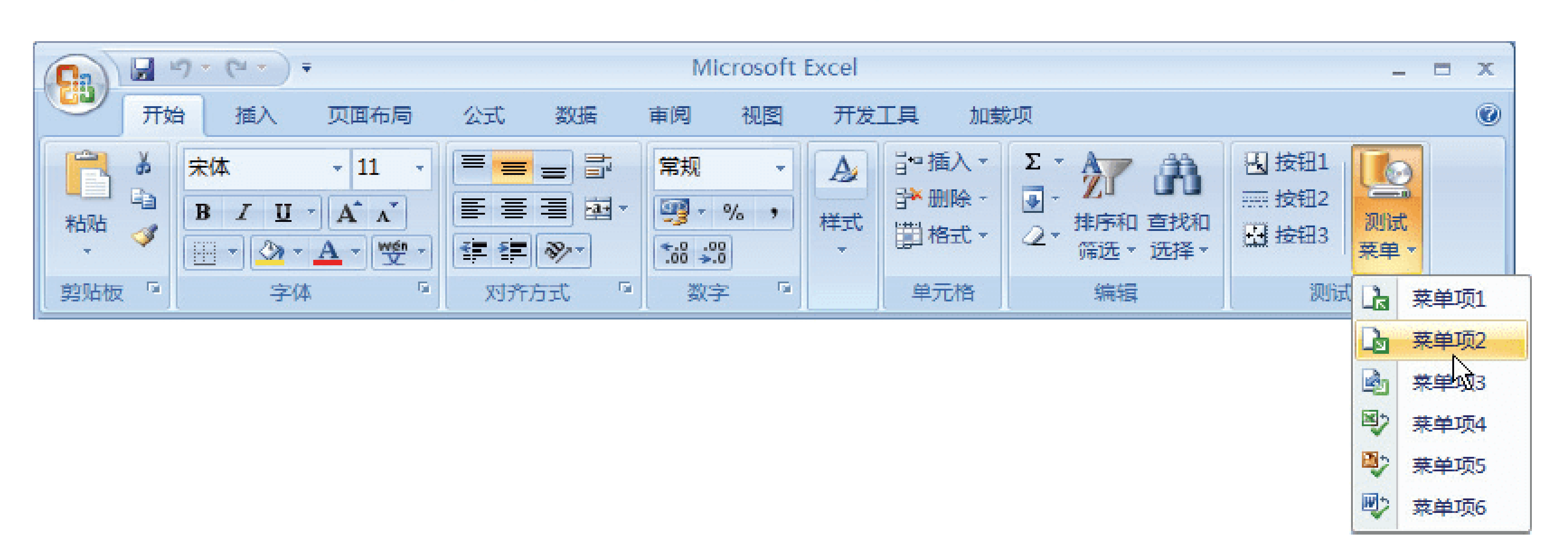


图 20-27　添加 Ribbonx 控件到内置选项卡

单击按钮或菜单项目，将调用工作簿中的宏。如果希望单击这些控件时能完成实际的工作，就需要在工作簿中定义对应的宏（例如，sub1、sub2、MenuSub1、MenuSub2 等）。

20.4.3　定义 Office 按钮

除了可以自定义功能区的选项卡外，用户还可以自定义 Office 按钮中的菜单项。例如，以下 XML 代码将在 Office 按钮中添加菜单项（分别添加 1 个菜单、1 个弹出菜单及其 6 个弹出菜单项、在系统已有的【打印】弹出菜单中添加 1 个菜单项）。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui" >
 <ribbon>
  <officeMenu>
      <button id="Button1" label="自定义菜单" onAction="sub1"
```

```
        imageMso="FindRelatedMenu" />
      <menu id="PopMenu" label="弹出菜单 u" imageMso="HappyFace"  >
        <button id="Button2" label="菜单项 1" onAction="sub2" imageMso =
        "FileCheckIn" />
        <button id="Button3" label="菜单项 2" onAction="sub3" imageMso =
        "FileCheckOut" />
        <button id="Button4" label="菜单项 3" onAction="sub4"
            imageMso="FileCheckOutDiscard" />
        <button id="Button5" label="菜单项 4" onAction="sub5"
            imageMso="FileCompatibilityChecker" />
        <button id="Button6" label="菜单项 5" onAction="sub6"
            imageMso="FileCompatibilityCheckerPowerPoint" />
        <button id="Button7" label="菜单项 6" onAction="sub7"
            imageMso="FileCompatibilityCheckerWord" />
      </menu>
     <splitButton idMso="FilePrintMenu">
        <menu>
          <button id="PrintButton" label="打印设置" onAction="sub8" imageMso=
          "FilePrint"
            description="由用户设置打印机参数"/>
        </menu>
      </splitButton>
    </officeMenu>
 </ribbon>
</customUI>
```

在 XML 中，使用元素<officeMenu>表示定义 Office 按钮菜单。

<splitButton idMso="FilePrintMenu">表示定义一个分离按钮（这里看起来像一个弹出式菜单），使用 idMso 表示引用内置的 FilePrintMenu 控件。

以上 XML 代码生成的 Office 按钮菜单如图 20-28 所示。

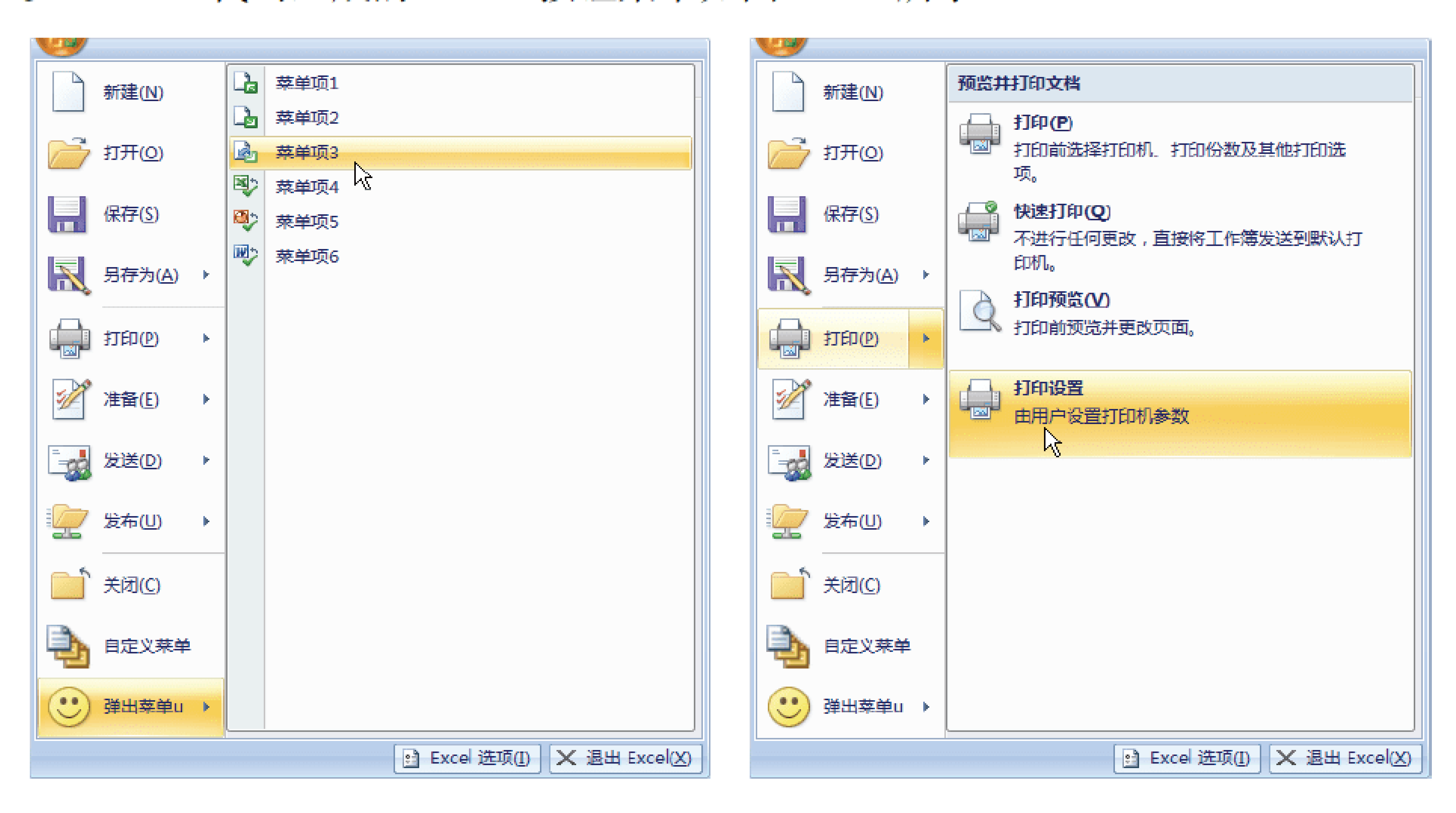


图 20-28　自定义 Office 按钮

20.4.4　RibbonX 控件回调函数实例

前面介绍的实例都没有使用 RibbonX 控件的回调函数。在实际应用中，很多地方都需要使用回调函数才能完成具体的工作。下面以实例演示回调函数的使用方法，创建类似 Excel 2007 功能区【开始】选项卡的【剪贴板】组中的【粘贴】分离按钮控件，当用户单击下方的按钮时将弹出下拉按钮列表，如图 20-29 左图所示。单击其中某个按钮后，上方显示的按钮将变为该按钮图标。鼠标指向按钮时，下方将显示提示信息，如图 20-29 右图所示。





图 20-29　新建 RibbonX 控件

具体的步骤如下：

（1）向 Excel 工作簿中添加以下自定义 RibbonX 控件的 XML 代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui"
    onLoad="rxcustomUI_onLoad">
  <ribbon>
    <tabs>
      <tab id="rxGoto"  label="跳转" >
       <group id="rxMoveCell" label="单元格移动" >
         <splitButton id="rxSplit"  size="large" >
             <button id="rxButton"   getImage="rxButton_getImage"
                  getLabel="rxButton_getLabel"
                  getSupertip="rxButton_getSupertip" onAction="rxButton_
                  onAction" />
            <menu id="rxMenu" >
                 <button id="rxMenuTop"  label="顶部"
                      imageMso="FillUp"
                      onAction="rxMenu_onAction" />
                 <button id="rxMenuLeft" label="左侧"
                      imageMso="FillLeft"
                      onAction="rxMenu_onAction" />
                <button id="rxMenuRight"  label="右侧"
                     imageMso="FillRight"
                     onAction="rxMenu_onAction" />
               <button id="rxMenuBottom"  label="底部"
                    imageMso="FillDown"
                    onAction="rxMenu_onAction" />
```

```
          </menu>
      </splitButton>
    </group>
   </tab>
  </tabs>
 </ribbon>
</customUI>
```

以上代码中，第 1 行语句中添加了 onLoad 属性，该属性设置系统装载自定义 RibbonX 时执行的回调函数名称，需要在 VBE 中编写名为 rxcustomUI_onLoad 的过程，具体代码如下：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui"
    onLoad="rxcustomUI_onLoad">
```

<splitButton>元素中包含 1 个按钮（<button>控件）和 1 个菜单（<menu>控件），在<menu>控件中添加 4 个按钮控件。<splitButton>元素中包含的按钮用来显示上一步操作过的按钮图标，在下拉菜单中选择不同按钮后，该按钮将自动更新其图标、提示文字和标签文字，因此，使用了 getImage、getLabel、getSupertip 3 个属性分别设置 3 个回调函数，具体代码如下：

```
<button id="rxButton"   getImage="rxButton_getImage"
    getLabel="rxButton_getLabel"
    getSupertip="rxButton_getSupertip" onAction="rxButton_onAction" />
```

而<menu>控件包含的 4 个按钮控件只需要使用 onAction 属性设置一个回调函数即可。

（2）将上述 XML 代码添加到 Excel 工作簿中后，在 Excel 2007 中打开该工作簿即可看到自定义的 RibbonX 控件，如图 20-29 所示。

（3）按组合键 Alt+F11 进入 VBE 环境。

（4）向工程中插入一个模块，在模块声明部分输入以下代码声明 2 个模块变量：

```
Dim moRibbon As IribbonUI     '模块变量，获取对 Ribbon 的引用
Dim str1 As String            '模块变量，保存当前按钮的状态
```

（5）当 Excel 2007 装载自定义 RibbonX 时，将调用 onLoad 属性设置的回调函数。在程序中只能通过该回调函数获取 RibbonX 对象的引用。具体代码如下：

```
Sub rxcustomUI_onLoad(ribbon As IRibbonUI)
    Set moRibbon = ribbon    '获取对 Ribbon 的引用
End Sub
```

（6）编写 getImage 属性设置的回调函数如下：

```
Sub rxButton_getImage(Control As IRibbonControl, ByRef returnedVal)
    If str1 = "" Then str1 = "Right"
    Select Case str1
        Case "Top"
            returnedVal = "FillUp"
        Case "Left"
            returnedVal = "FillLeft"
```

```
        Case "Right"
            returnedVal = "FillRight"
        Case "Bottom"
            returnedVal = "FillDown"
    End Select
End Sub
```

回调函数的参数可参见本章 20.2.4 节中的介绍。以上代码通过模块变量 str1，判断当前使用过的按钮，再使用参数 returnedVal 返回对应按钮的图像 ID（使用内置图像）。

（7）接着使用类似的方法编写 getLabel 属性设置的回调函数，用来修改按钮的显示文本，具体代码如下：

```
Sub rxButton_getLabel(ByRef Control As IRibbonControl, ByRef ReturnValue
As Variant)
    If str1 = "" Then str1 = "Right"
    Select Case str1
        Case "Top"
            ReturnValue = "顶部"
        Case "Left"
            ReturnValue = "左侧"
        Case "Right"
            ReturnValue = "右侧"
        Case "Bottom"
            ReturnValue = "底部"
    End Select
End Sub
```

（8）编写 getSupertip 属性设置的回调函数，用来设置按钮的提示文字，具体代码如下：

```
Sub rxButton_getSupertip(ByRef Control As IRibbonControl, ByRef ReturnValue
As Variant)
    If str1 = "" Then str1 = "Right"
    Select Case str1
        Case "Top"
            ReturnValue = "移动到区域顶部"
        Case "Left"
            ReturnValue = "移动到区域左侧"
        Case "Right"
            ReturnValue = "移动到区域右侧"
        Case "Bottom"
            ReturnValue = "移动到区域底部"
    End Select
End Sub
```

（9）当用户单击元素<menu>时，执行 onAction 属性设置的回调函数，具体代码如下：

```
Sub rxMenu_onAction(Control As IRibbonControl)
    str1 = Mid$(Control.ID, 7)
    moRibbon.InvalidateControl "rxButton"   '更新按钮控件
    DoGoto str1
End Sub
```

以上代码首先获取单击菜单容器中的控件（为按钮控件）的 ID，因 ID 前面有前缀 rxMenu，所以使用 Mid 函数从第 7 个字符开始取子串。

然后使用 IribbonUI 对象的 InvalidateControl 方法更新按钮控件。这时，按钮控件将执行 getImage、getLabel、getSupertip 属性设置的回调函数。

最后调用过程 DoGoto 移动单元格的位置。

（10）过程 DoGoto 的 VBA 代码如下，根据参数的值确定执行的具体操作。

```
Private Sub DoGoto(ByVal sStyle As String)
    Select Case sStyle
        Case "Top"
            ActiveCell.End(xlUp).Select
        Case "Left"
            ActiveCell.End(xlToLeft).Select
        Case "Right"
            ActiveCell.End(xlToRight).Select
        Case "Bottom"
            ActiveCell.End(xlDown).Select
    End Select
End Sub
```

（11）单击<button>控件时，将执行 onAction 属性设置的回调函数，具体代码如下：

```
Sub rxButton_onAction(Control As IRibbonControl)
    DoGoto str1
End Sub
```

代码编写完成以后，在工作表中单击选择一个单元格，再单击【跳转】选项卡【单元格移动】组中的按钮，可在当前区域中快速移动单元格，定位到区域的边界上。

第 21 章　使用 CommandBars

CommandBars 对象代表 Excel 应用程序中命令栏的 CommandBar 对象的集合。CommandBar 对象代表 Excel 应用程序中的一个命令栏，指菜单、快捷菜单、工具栏等。开发人员可创建 CommandBar 对象，将应用程序的功能绑定到这些对象上，方便用户调用。

在 Excel 2003 及以前版本中，自定义菜单或工具栏将以常规方式显示。在 Excel 2007 中，使用功能区代替了菜单和工具栏。用户自定义的菜单、工具栏将显示在【加载项】选项卡中。

21.1　CommandBar 对象

菜单栏和工具栏是用户与 Excel 进行交互的工具，在 Excel 中，将菜单栏、工具栏和快捷菜单合并为一种功能，称为命令栏，都放置到 CommandBars 集合中。

21.1.1　CommandBars 简介

CommandBars 集合包含 Excel 中的所有命令栏，即是所有 CommandBar 对象的集合，而每个 CommandBar 对象都有一个 CommandBarControls 集合，该集合由多个 CommandBarControl 对象组成。对象模型结构如图 21-1 所示。

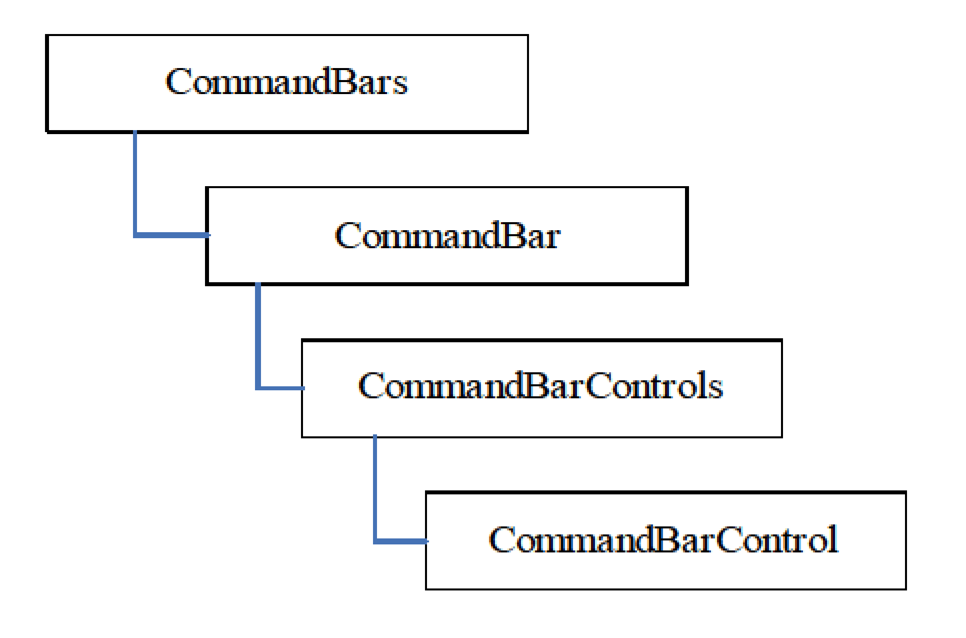


图 21-1　CommandBar 对象模型

21.1.2　CommandBars 对象常用属性

- ActiveMenuBar 属性，返回一个 CommandBar 对象。该对象代表 Excel 应用程序中

的活动菜单栏。

- Count 属性：返回 Excel 应用程序中的命令栏数量。
- DisplayTooltips 属性：设置是否显示命令栏控件的提示信息，该属性若为 True，则显示提示信息，若为 False，则不显示提示信息。
- Item 属性：返回 CommandBars 集合中的 CommandBar 对象。
- LargeButtons 属性：若为 True，则工具栏按钮比常规尺寸要大，若为 False，工具栏按钮显示为常规尺寸。

21.1.3 CommandBars 对象常用方法

使用 CommandBars 集合对象的方法可完成添加命令栏，在集合中查询命令按钮等操作。

1. Add方法

使用该方法新建一个命令栏并添加到命令栏集合，返回 CommandBar 对象。其格式为：

```
CommandBars.Add(Name, Position, MenuBar, Temporary)
```

各参数的含义如下所述。

- Name：为新命令栏的名称。如果忽略该参数，则为默认名称（例如，Custom 1）。
- Position：设置新命令栏的位置或类型。
- MenuBar：值为 True 时，表示将以新命令栏替换活动菜单栏。默认值为 False。
- Temporary：值为 True 时，表示使新命令栏成为临时命令栏。临时命令栏将在关闭容器应用程序时删除。默认值为 False。

注意：对 CommandBars 对象集合使用 Add 方法，是添加新的菜单栏或工具栏，并没有创建具体的菜单或命令按钮。

2. ExecuteMso方法

该方法执行由 idMso 参数标识的控件。例如，以下代码执行【复制】按钮的功能，将活动单元格的内容复制到剪贴板中：

```
Application.CommandBars.ExecuteMso("Copy")
```

3. GetImageMso方法

该方法返回由 idMso 参数标识的控件图像（缩放到指定的宽度和高度尺寸）的 IPictureDisp 对象。其语法格式如下：

```
表达式.GetImageMso(idMso, Width, Height)
```

各参数的含义如下所述。

- idMso：控件的标识符。

- Width：图像的宽度。
- Height：图像的高度。

例如，以下代码将【粘贴】按钮的图像设置到工作表 Sheet2 中的 OLE 对象中（如插入在工作表中的【图像】或【命令按钮】等 ActiveX 控件）：

```
Set Worksheets("sheet2").OLEObjects(1).Object.Picture = _
    Application.CommandBars.GetImageMso("Paste", 32, 32)
```

21.1.4　CommandBar 对象常用属性

CommandBar 对象是 CommandBars 集合中的成员，用 CommandBars(index)可返回一个 CommandBar 对象，该对象对应到具体的某个菜单栏或工具栏。以下是 CommandBar 对象的常用属性。

- Name 属性：为对象的名称。对于 Excel 的内置命令栏，Name 属性返回该命令栏的英文名称。另外，用 NameLocal 属性可返回其本地名称。
- NameLocal 属性：返回内置命令栏的本地名称，无论有效命令栏列表显示在容器应用程序的哪个位置，内置命令栏的本地名都将显示在标题栏中。如果改变了一个自定义命令栏的 LocalName 属性值，那么该命令栏的 Name 属性值也将改变，反之亦然。
- Controls 属性：返回一个 CommandBarControls 对象。该对象代表指定命令栏或弹出式控件中的所有控件集合。
- Enabled 属性：设置命令栏是否要用。如果将该属性设置为 True，那么命令栏的名称将出现在有效命令栏列表中。
- Type 属性：返回命令栏的类型。可设置为 3 个值：msoBarTypeNormal（值为 0）表示工具栏；msoBarTypeMenuBar（值为 1）表示菜单栏；msoBarTypePopup（值为 2）表示快捷菜单。
- Visible 属性：设置命令栏是否可见。值为 True 时，表示可见，值为 False 时，则表示不可见。新建自定义命令栏的 Visible 属性的默认值为 False。只有先将命令栏的 Enabled 属性设置为 True，才可将其 Visible 属性设置为 True。

21.1.5　CommandBar 对象常用方法

CommandBar 对象提供了 4 个方法，可分别用来删除命令栏、查找指定控件、重置命令栏、显示快捷菜单等操作。

1．Delete方法

使用该方法从集合中删除 CommandBar 对象。例如，以下命令将删除所有不可见的自定义命令栏。

```
foundFlag = False
delBars = 0
```

```
For Each bar In CommandBars
    If (bar.BuiltIn = False) And (bar.Visible = False) Then
        bar.Delete
        foundFlag = True
        delBars = delBars + 1
    End If
Next bar
If Not foundFlag Then
    MsgBox "没有自定义命令栏被删除！"
Else
    MsgBox "共删除 " & delBars & " 个自定义命令栏！"
End If
```

2．FindControl方法

使用该方法获取一个符合指定条件的 CommandBarControl 对象。其语法格式如下：

```
表达式.FindControl(Type, Id, Tag, Visible, Recursive)
```

各参数的含义如下所述。

- Type：要查找控件的类型。
- ID：要查找控件的标识符。
- Tag：要查找控件的标记值。
- Visible：该参数如果为 True，则表示只能在搜索中包含可见的命令栏控件。默认值为 False。可见的命令栏包括所有可见的工具栏和执行 FindControl 方法时打开的所有菜单。
- Recursive：该参数如果为 True，则表示将在命令栏及其所有弹出式子工具栏中查找。

3．Reset方法

该方法将内置命令栏重置为其默认配置。

4．ShowPopup方法

该方法将指定的命令栏作为快捷菜单，在指定坐标或当前光标位置显示。

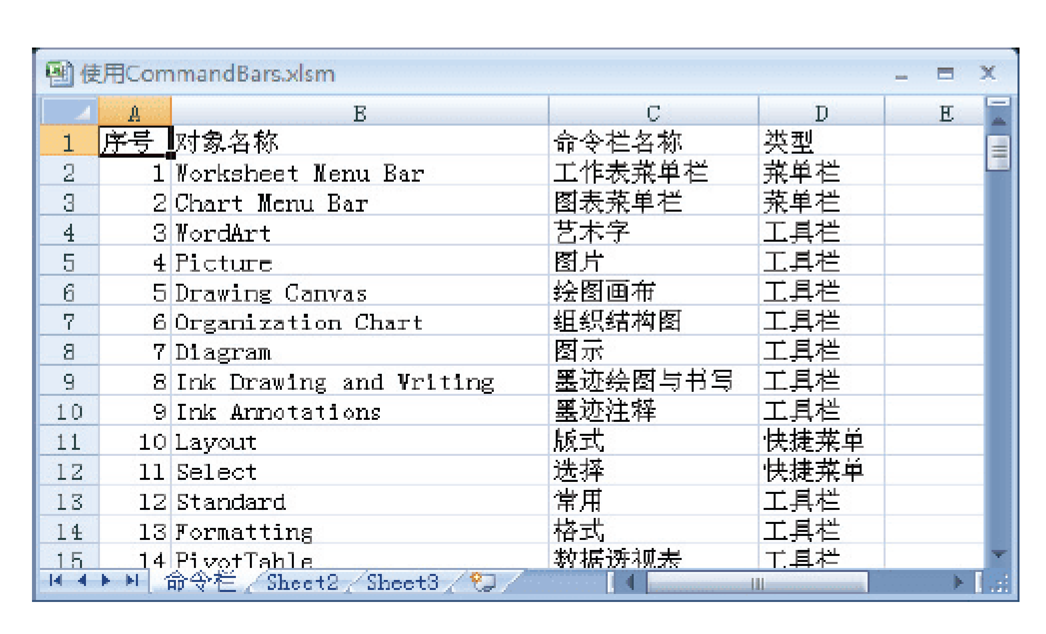
21.1.6 列出命令栏

通过遍历 CommandBars 集合对象可显示出 Excel 内置的命令栏。具体代码如下：

```
Sub 显示所有命令栏()
    Dim cBar As CommandBar, i As Integer
    With Worksheets("命令栏")
        i = 2
        For Each cBar In CommandBars
            .Cells(i, 1) = cBar.Index
```

```
            .Cells(i, 2) = cBar.Name
            .Cells(i, 3) = cBar.NameLocal
            Select Case cBar.Type
                Case msoBarTypeNormal
                    .Cells(i, 4) = "工具栏"
                Case msoBarTypeMenuBar
                    .Cells(i, 4) = "菜单栏"
                Case msoBarTypePopup
                    .Cells(i, 4) = "快捷菜单"
            End Select
            i = i + 1
        Next
        .Cells.Columns.AutoFit
    End With
End Sub
```

执行以上代码，在工作表“命令栏”中可看到 Excel 系统所有命令栏的信息，如图 21-2 所示。



使用CommandBars.xlsm

序号	对象名称	命令栏名称	类型
1	Worksheet Menu Bar	工作表菜单栏	菜单栏
2	Chart Menu Bar	图表菜单栏	菜单栏
3	WordArt	艺术字	工具栏
4	Picture	图片	工具栏
5	Drawing Canvas	绘图画布	工具栏
6	Organization Chart	组织结构图	工具栏
7	Diagram	图示	工具栏
8	Ink Drawing and Writing	墨迹绘图与书写	工具栏
9	Ink Annotations	墨迹注释	工具栏
10	Layout	版式	快捷菜单
11	Select	选择	快捷菜单
12	Standard	常用	工具栏
13	Formatting	格式	工具栏
14	PivotTable	数据透视表	工具栏

命令栏　Sheet2　Sheet3

图 21-2　Excel 所有命令栏

21.2　CommandBarControl 对象

CommandBarControl 对象为具体的命令按钮或菜单项，该对象是 CommandBarControls 集合中的成员。每个 CommandBar 对象包含一个 CommandBarControls 集合对象。

21.2.1　CommandBarControls 集合对象

使用 CommandBar 对象的 Controls 属性，可获取一个代表命令栏上的所有控件的 CommandBarControls 集合对象。

CommandBarControls 集合对象具有通用集合对象的属性和方法。通过 Add 方法可新建一个 CommandBarControl 对象，并将其添加到指定命令栏上的控件集合中。语法格式如下：

```
expression.Add(Type, Id, Parameter, Before, Temporary)
```

各参数的含义如下所述。

- ❑ Type：设置添加到指定命令栏的控件类型。可以为下列 MsoControlType 常量之一：msoControlButton、msoControlEdit、msoControlDropdown、msoControlComboBox 或 msoControlPopup。
- ❑ Id：用来指定内置控件的整数。如果该参数为 1，或者忽略该参数，将在命令栏中添加一个空的指定类型的自定义控件。
- ❑ Parameter：设置参数信息，对于内置控件，该参数用于容器应用程序运行命令。对于自定义控件，可以使用该参数向 VBA 过程传递信息，或用其存储控件信息。
- ❑ Before：设置位置信息，表示新控件在命令栏上位置的数字，新控件将插入到该位置控件之前。如果忽略该参数，控件将添加到指定命令栏的末端。
- ❑ Temporary：值为 True 时，表示将使新命令栏成为临时命令栏。临时命令栏在关闭容器应用程序时删除，其默认值为 False。

21.2.2 CommandBarControl 对象

CommandBarControl 对象为具体的命令按钮或菜单项，对象是 CommandBarControls 集合中的成员。下面简单介绍 CommandBarControl 对象的常用属性和方法。

1. CommandBarControl对象常用属性

- ❑ BeginGroup 属性：设置为 True 时会在控件前显示一个分隔条；
- ❑ BuiltIn 属性：若控件为内置控件，则该属性的值为 True；
- ❑ Caption 属性：为控件上显示的文本；
- ❑ Enabled 属性：设置为 True 时可以单击该控件；
- ❑ FaceID 属性：代表图形图像的数字；
- ❑ ID 属性：代表内置控件的代码编号；
- ❑ OnAction 属性：指定单击控件时执行的 VBA 过程名称；
- ❑ State 属性：确定控件是否呈现按下状态，仅用于 CommandBarButton 控件；
- ❑ Style 属性：确定控件显示时是否包含标题或图像，只能用于 CommandBarButton 控件和 CommandBarComboBox 控件；
- ❑ ToolTipText 属性：控件的提示文本，当鼠标移动到控件上方时呈现；
- ❑ Type 属性：表示控件类型。

2. CommandBarControl对象常用方法

- ❑ Copy 方法：将一个命令栏控件复制到已有的命令栏中；
- ❑ Delete 方法：将 CommandBarControl 对象从其集合中删除；
- ❑ Execute 方法：运行分配给指定 CommandBarControl 控件的过程或内置命令；
- ❑ Move 方法：将指定的 CommandBarControl 移动到已有的命令栏；
- ❑ SetFocus 方法：将键盘的焦点移到指定 CommandBarControl。如果该控件被禁用或

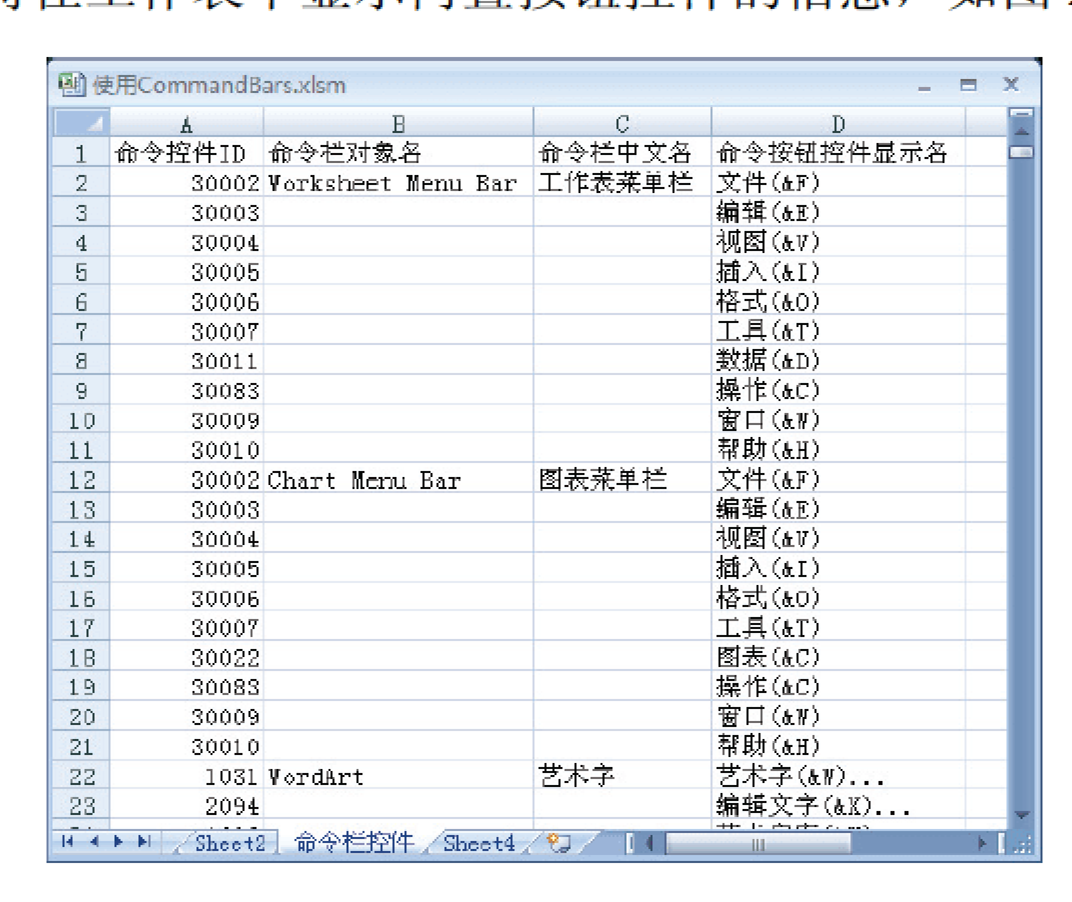
不可见，那么此方法将失败。

21.2.3　列出内置命令栏控件

通过 CommandBars 集合对象和 CommandBarControls 集合对象可遍历 Excel 内置的命令栏控件。具体代码如下：

```
Sub 显示命令栏控件()
    Dim cb As CommandBar, cbc As CommandBarControl
    Dim i As Long, j As Long
    With Worksheets("命令栏控件")
        i = 2
        For Each cb In CommandBars                   '循环处理每个命令栏
            .Cells(i, 2) = cb.Name                   '获取命令栏名称
            .Cells(i, 3) = cb.NameLocal              '获取命令栏中文名称
            For Each cbc In cb.Controls              '循环处理每个命令栏中的控件按钮
                .Cells(i, 1) = cbc.ID                '控件按钮 ID
                .Cells(i, 4) = cbc.Caption           '控件按钮显示文字
                i = i + 1
            Next
        Next
        .Cells.Columns.AutoFit
    End With
End Sub
```

运行以上代码，将在工作表中显示内置按钮控件的信息，如图 21-3 所示。



使用CommandBars.xlsm

	A	B	C	D
1	命令控件ID	命令栏对象名	命令栏中文名	命令按钮控件显示名
2	30002	Worksheet Menu Bar	工作表菜单栏	文件(&F)
3	30003			编辑(&E)
4	30004			视图(&V)
5	30005			插入(&I)
6	30006			格式(&O)
7	30007			工具(&T)
8	30011			数据(&D)
9	30083			操作(&C)
10	30009			窗口(&W)
11	30010			帮助(&H)
12	30002	Chart Menu Bar	图表菜单栏	文件(&F)
13	30003			编辑(&E)
14	30004			视图(&V)
15	30005			插入(&I)
16	30006			格式(&O)
17	30007			工具(&T)
18	30022			图表(&C)
19	30083			操作(&C)
20	30009			窗口(&W)
21	30010			帮助(&H)
22	1031	WordArt	艺术字	艺术字(&W)...
23	2094			编辑文字(&X)...

Sheet2　命令栏控件　Sheet4

图 21-3　内置命令按钮控件

21.3　自定义菜单

Excel 2003 及以前版本采用菜单驱动，用户通过菜单命令来完成大部分的工作。开发

人员还可使用 VBA 代码自定义菜单。在 Excel 2007 中，除快捷菜单能显示外，其余菜单栏都是隐藏状态。为了和以前版本兼容，可使用 VBA 代码访问这些菜单栏。

21.3.1 菜单的构成

Excel 有两个内置的菜单栏：

- ❑ 工作表菜单栏，对象名为 Worksheet Menu Bar，如图 21-4 所示为工作表菜单栏。
- ❑ 图表菜单栏，对象名为 Chart Menu Bar。

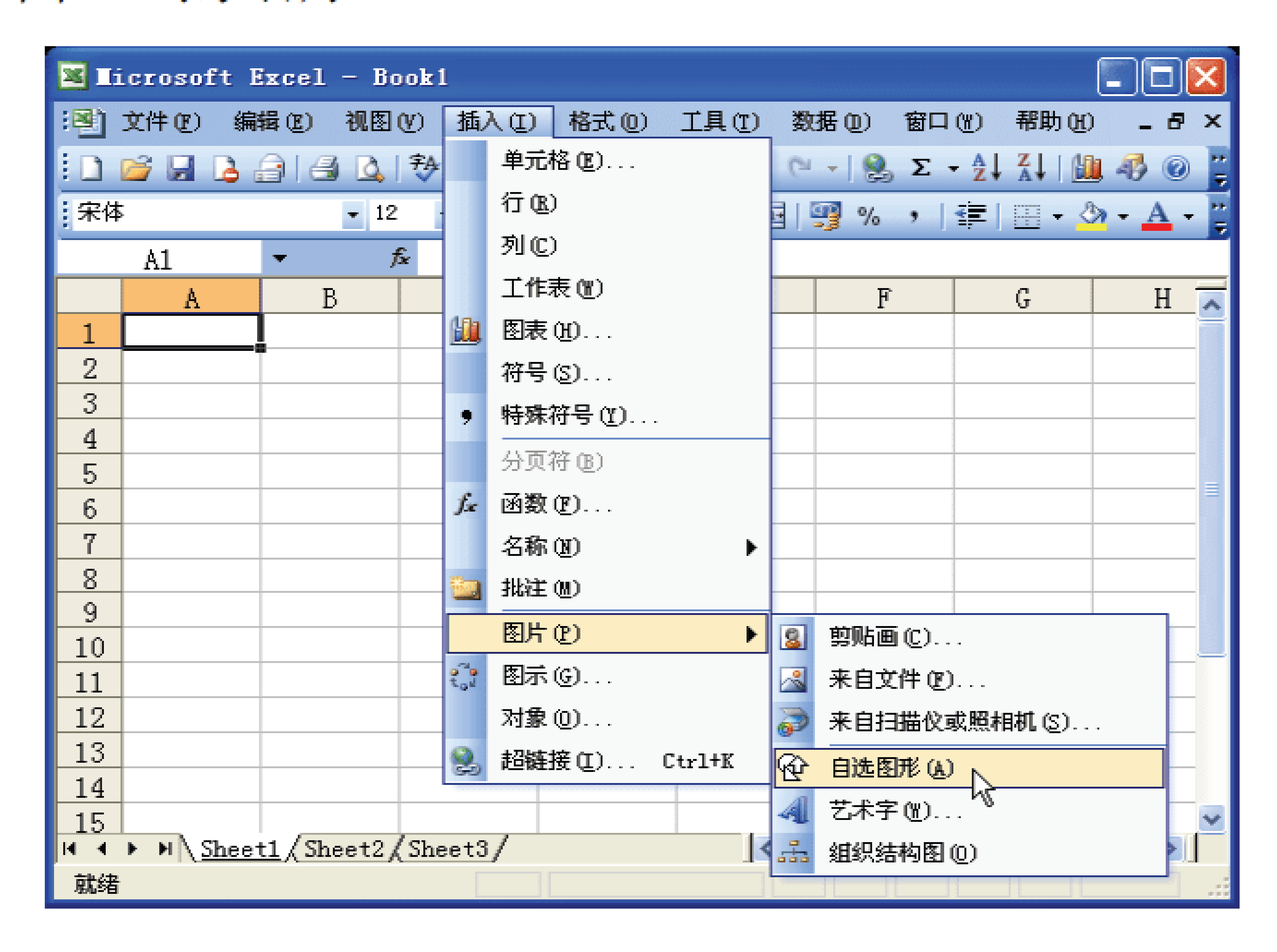


图 21-4 Excel 2003 的菜单

菜单栏属于 CommandBars 集合，可以通过菜单栏名称或索引值对其进行引用，例如，以下代码可引用工作表菜单：

```
CommandBars("Worksheet Menu Bar")
```

或：

```
CommandBars(1)
```

菜单栏中的每一个下拉菜单（如【视图】、【插入】等）为一个 CommandBarControl 对象，下拉菜单中的菜单项（如【插入】下拉菜单中的【单元格】等）也是一个 CommandBarControl 对象，通过属性 CommandBarControl 对象的 Type 属性来区分是下拉菜单，还是菜单项。将 Type 属性设置为以下值即可。

- ❑ msoControlButton：值为 1，表示对象为菜单项；
- ❑ msoControlPopup：值为 10，表示对象为下拉菜单。

21.3.2 创建新菜单

使用 VBA 代码可以方便地向 Excel 菜单栏中添加新菜单。使用 CommandBarControls

集合对象的 Add 方法，可在菜单中添加新的菜单。Add 方法可以指定控件的类型、内置控件的 ID 号、位置。还可以指定新添加的菜单是否为一个临时控件，如果是一个临时控件，则在关闭 Excel 时会自动删除该菜单。

在新建菜单时，可以指定新菜单的位置。如果不指定，则会在菜单工具栏末尾添加新菜单。

一般定义一个对象变量，用来保存 Add 方法创建好的新菜单，在程序中即可使用该对象变量设置菜单的各种属性。使用 Caption 属性指定新菜单的名称，使用 OnAction 属性指定单击菜单后的行为。

为了避免在菜单栏中重复加入同样的自定义菜单，在新建菜单前一般先使用 FindControl 方法查找，如果已存在该菜单，可将其删除。

下面的实例在工作表菜单栏右侧增加一个下拉菜单，在该下拉菜单中添加 4 个菜单项，分别用来选择指定的区域。在 Excel 2003 中，自定义菜单显示如图 21-5 所示效果。

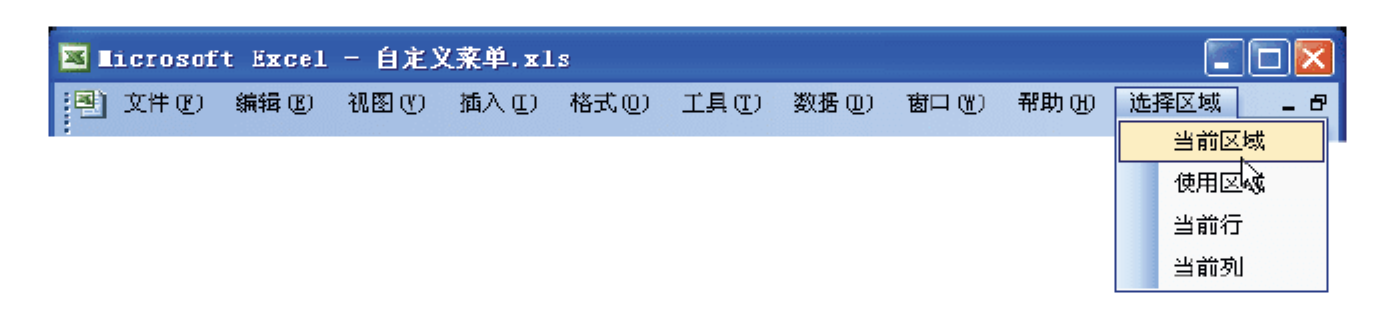


图 21-5　Excel 2003 中显示自定义菜单

在 Excel 2007 中，自定义菜单将显示在【加载项】选项卡中，如图 21-6 所示。

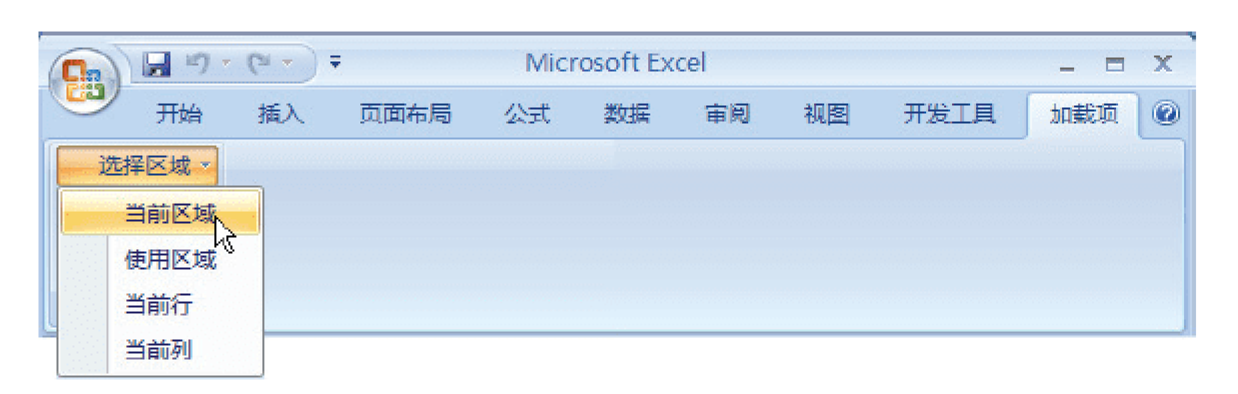


图 21-6　Excel 2007 中显示自定义菜单

创建图 21-5 所示自定义菜单的具体步骤如下：

（1）按组合键 Alt+F11 切换到 VBE 环境。

（2）向工程中增加一个模块，在模块中编写以下代码，创建自定义菜单：

```
Sub AddMenu()
    Dim myMenubar As CommandBar                    '声明变量，保存工作表菜单栏
    Dim myMenu As CommandBarControl
    Dim myMenuitem As CommandBarControl
    Set myMenubar = CommandBars(1)                 '获取工作表菜单栏

    Set myMenu = myMenubar.FindControl(Tag:="mySelect")
    If Not (myMenu Is Nothing) Then                '存在该菜单
        myMenu.Delete                              '删除该菜单
    End If
```

```
    Set myMenu = myMenubar.Controls.Add(Type:=msoControlPopup)
    With myMenu
        .Caption = "选择区域"                    '设置显示文本
        .Tag = "mySelect"                      '设置一个标记，以方便查找
    End With

    Set myMenuitem = myMenu.Controls.Add(Type:=msoControlButton)
    With myMenuitem
        .Caption = "当前区域"
        .Tag = "SelectCR"                      '设置标记
        .OnAction = "mnu_CurrentRegion"        '设置选择该菜单时调用的子过程
    End With

    Set myMenuitem = myMenu.Controls.Add(Type:=msoControlButton)
    With myMenuitem
        .Caption = "使用区域"
        .Tag = "SelectUR"
        .OnAction = "mnu_UsedRange"
    End With

    Set myMenuitem = myMenu.Controls.Add(Type:=msoControlButton)
    With myMenuitem
        .Caption = "当前行"
        .Tag = "SelectRow"
        .OnAction = "mnu_Row"
    End With

    Set myMenuitem = myMenu.Controls.Add(Type:=msoControlButton)
    With myMenuitem
        .Caption = "当前列"
        .Tag = "SelectCol"
        .OnAction = "mnu_Column"
    End With
End Sub
```

以上代码首先获取工作表菜单栏的引用，再使用 FindControl 方法查找当前菜单栏中是否已有标记为 mySelect 的控件（创建菜单时不方便指定其 ID 值，可通过设置 Tag 值来识别按钮控件），若存在该控件就调用 Delete 方法将其删除，然后使用 Add 方法创建新的下拉菜单，最后再在该下拉菜单中创建 4 个菜单项，分别为这 4 个菜单项的名称、调用的子过程等参数。

（3）为每个菜单项编写事件处理子过程，用来选择相应的单元格区域。具体代码如下：

```
Sub mnu_CurrentRegion()                        '选择当前区域
    ActiveCell.CurrentRegion.Select
End Sub
Sub mnu_UsedRange()                            '选择工作表使用区域
```

```
    On Error Resume Next
    ActiveSheet.UsedRange.Select
End Sub
Sub mnu_Row()                                       '选择当前单元格所在行
    ActiveCell.EntireRow.Select
End Sub
Sub mnu_Column()                                    '当前单元格所在列
    ActiveCell.EntireColumn.Select
End Sub
```

编写好以上代码后，执行 AddMenu 子过程，即可创建“选择区域”下拉菜单。激活一个工作表，选择菜单中的命令就可选择指定的区域。

（4）若需要 Excel 工作簿打开时就将自定义菜单添加到菜单栏中，可以在工作簿的 Open 事件中编写以下代码：

```
Private Sub Workbook_Open()
    AddMenu
End Sub
```

（5）在关闭工作簿之前，应该将自定义菜单删除，以免自定义菜单出现在以后创建的 Excel 工作簿中，具体代码如下：

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    On Error Resume Next
    Set myMenu = CommandBars(1).FindControl(Tag:="mySelect")
    If Not (myMenu Is Nothing) Then                 '存在该菜单
        myMenu.Delete                               '删除该菜单
    End If
End Sub
```

21.4　自定义快捷菜单

Windows 操作系统的一个便捷功能就是提供了功能强大的右键快捷菜单。在任何一个对象上按鼠标右键，就会弹出一个与所选当前对象相关的菜单，菜单中列出了一组针对当前对象的操作。Excel 应用程序提供了很多内置快捷菜单，以方便用户编辑操作工作表中的数据，开发人员还可根据需要自定义快捷菜单。

21.4.1　内置快捷菜单

Excel 的快捷菜单有很多，在不同的对象上单击鼠标右键将弹出不同的快捷菜单列表。例如，在单元格上单击鼠标右键弹出的快捷菜单如图 21-7 所示。在工作表列标签上单击鼠标右键弹出的快捷菜单如图 21-8 所示，在行标签上单击右键弹出的快捷菜单如图 21-9 所示。

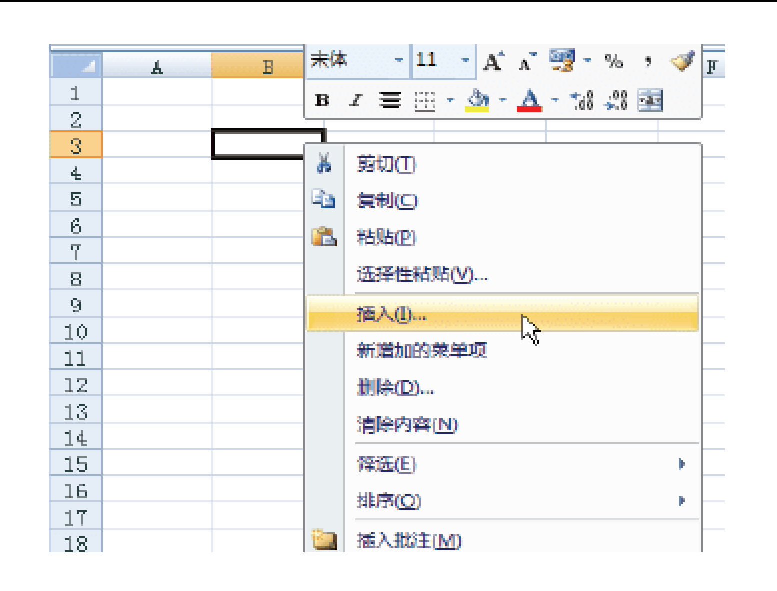


图 21-7　单元格快捷菜单

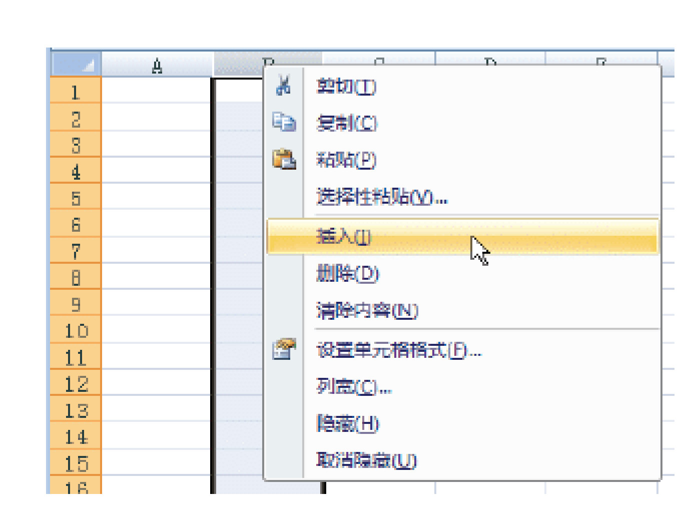


图 21-8　列标签快捷菜单

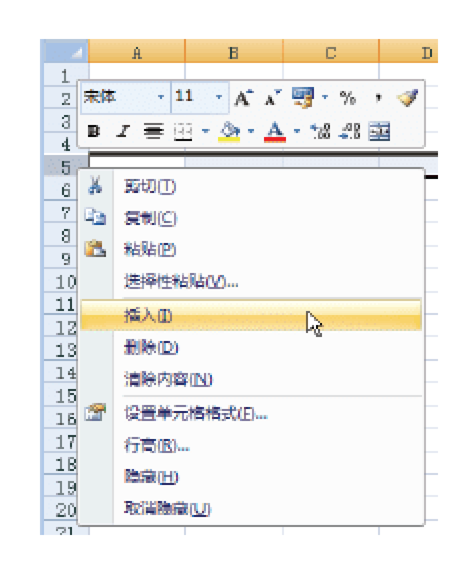


图 21-9　行标签快捷菜单

使用以下命令可列出 Excel 2007 中内置的快捷菜单：

```
Sub 显示快捷菜单()
    Dim cb As CommandBar, cbc As CommandBarControl
    Dim i As Long, j As Long
    With Worksheets("内置快捷菜单")
        i = 2
        For Each cb In CommandBars                    '循环处理每个命令栏
            If cb.Type = msoBarTypePopup Then         '处理快捷菜单
                .Cells(i, 2) = cb.Name                '获取命令栏名称
                .Cells(i, 3) = cb.NameLocal           '获取命令栏中文名称
                For Each cbc In cb.Controls           '循环处理每个命令栏中的控件按钮
                    .Cells(i, 1) = cbc.ID             '控件按钮 ID
                    .Cells(i, 4) = cbc.Caption        '控件按钮显示文字
                    i = i + 1
                Next
            End If
        Next
        .Cells.Columns.AutoFit
    End With
End Sub
```

执行以上代码，在工作表“内置快捷菜单”中将显示快捷菜单的相关信息，如图 21-10 所示，在图中显示了“单元格”快捷菜单中的命令。

自定义快捷菜单.xls

	A	B	C	D	E
33	21	Cell	单元格	剪切(&T)	
34	19			复制(&C)	
35	22			粘贴(&P)	
36	755			选择性粘贴(&V)...	
37	295			插入单元格(&E)...	
38	292			删除(&D)...	
39	3125			清除内容(&N)	
40	31402			筛选(&E)	
41	31435			排序(&O)	
42	2031			插入批注(&M)	
43	1592			删除批注(&M)	
44	1593			显示/隐藏批注(&O)	
45	855			设置单元格格式(&F)...	
46	1966			从下拉列表中选择(&K)...	
47	1614			显示拼音字段(&S)	
48	13380			命名单元格区域(&R)...	
49	1576			超链接(&H)...	
50	1577			编辑超链接(&H)...	
51	1015			打开超链接(&O)...	
52	3626			取消超链接(&R)	
53	11299			展开到明细(&X)	
54	31595			其他操作(&I)	
55	178			全屏显示(&U)	
56	21	Column	列	剪切(&T)	
57	19			复制(&C)	

内置快捷菜单　Sheet2　Sheet3

图 21-10　内置快捷菜单

21.4.2　创建快捷菜单

除了 Excel 内置的快捷菜单外，用户可根据需要创建新的快捷菜单。在工作表中拖动选择一个单元格区域，右击将弹出如图 21-11 所示的自定义快捷菜单。下面演示创建该快捷菜单的方法。

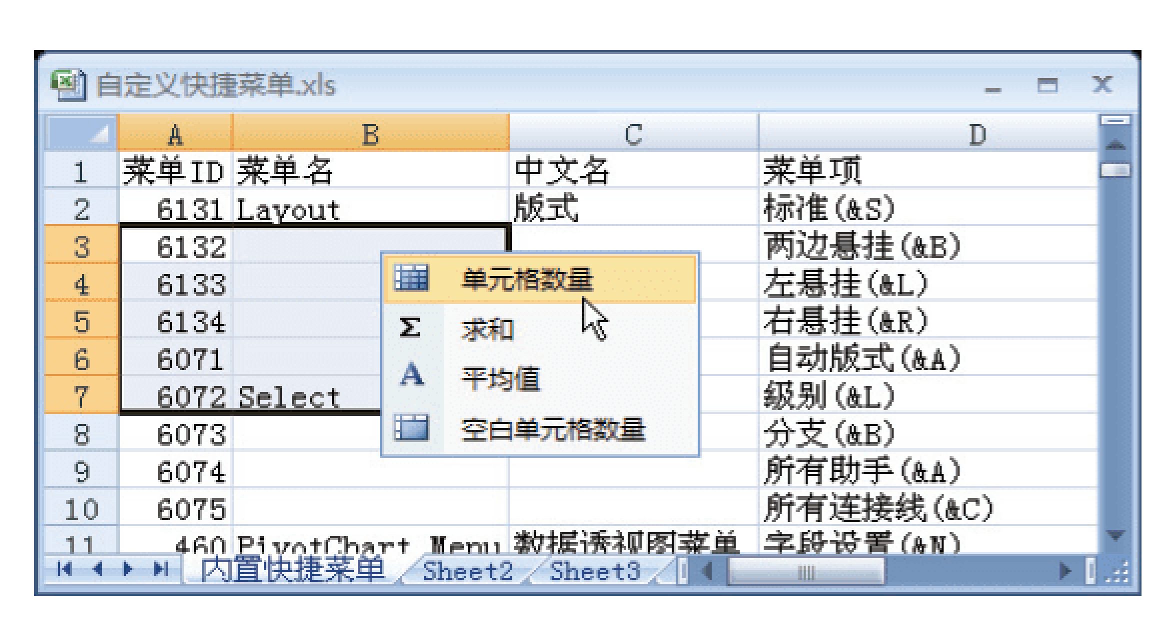


图 21-11　自定义快捷菜单

（1）在 VBE 中插入一个模块，在模块中编写创建快捷菜单的代码，具体代码如下：

```
Sub CreatePopup()
    Dim myBar As CommandBar
    Dim myItem As CommandBarControl

    DeletePopup                                  '若已存在该菜单,则删除

    Set myBar = CommandBars.Add _
      (Name:="StatPopup", Position:=msoBarPopup, Temporary:=False)
                                                 '创建快捷菜单栏

    Set myItem = myBar.Controls.Add(Type:=msoControlButton)
                                                 '添加菜单项
    With myItem
        .Caption = "单元格数量"                  '菜单名字
        .OnAction = "pm_Count"                   '调用过程
        .FaceId = 800                            '设置图标
    End With

    Set myItem = myBar.Controls.Add(Type:=msoControlButton)
    With myItem
        .Caption = "求和"
        .OnAction = "pm_Sum"
        .FaceId = 226
    End With

    Set myItem = myBar.Controls.Add(Type:=msoControlButton)
    With myItem
        .Caption = "平均值"
```

```
        .OnAction = "pm_Avg"
        .FaceId = 401
    End With

    Set myItem = myBar.Controls.Add(Type:=msoControlButton)
    With myItem
        .Caption = "空白单元格数量"
        .OnAction = "pm_BlankCell"
        .FaceId = 798
    End With
End Sub
```

以上代码首先调用 DeletePopup 过程删除已经存在的名为 StatPopup 的快捷菜单。再使用以下语句创建一个新的快捷菜单栏：

```
Set myBar = CommandBars.Add _
  (Name:="StatPopup", Position:=msoBarPopup, Temporary:=False)
```

在 Add 方法中，设置 Name 参数为 StatPopup、Position 参数为 msoBarPopup 表示创建弹出式快捷菜单。

最后向新建的快捷菜单中添加 4 个菜单项，并分别设置菜单项的名称和调用的过程。

（2）上面的代码调用 DeletePopup 过程删除已存在的名为 StatPopup 的快捷菜单，具体代码如下：

```
Sub DeletePopup()                                          '删除快捷菜单
    On Error Resume Next
    CommandBars("StatPopup").Delete
End Sub
```

（3）接着编写各菜单项调用的过程，具体代码如下：

```
Sub pm_Count()
    MsgBox "选择区域共有" & Selection.Cells.Count & "个单元格。"
End Sub
Sub pm_Sum()
    Dim rng1 As Range, t As Long
    For Each rng1 In Selection.Cells
        t = t + Val(rng1.Value)
    Next
    MsgBox "选择区域数值和为：" & t
End Sub
Sub pm_Avg()
    Dim rng1 As Range, t As Long
    For Each rng1 In Selection.Cells
        t = t + Val(rng1.Value)
    Next
    MsgBox "选择区域平均值为：" & t / Selection.Cells.Count
End Sub
Sub pm_BlankCell()
    Dim rng1 As Range, t As Long
    For Each rng1 In Selection.Cells
```

```
        If IsEmpty(rng1.Value) Then t = t + 1
    Next
    MsgBox "选择区域空白单元格有" & t & "个。"
End Sub
```

通过以上代码，就完成了快捷菜单的定义过程。

（4）快捷菜单在用户单击鼠标右键时将弹出，因此，在需要使用自定义快捷菜单的区域捕获单击右键的事件，在该事件中编写代码，显示出快捷菜单。

```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Excel.Range, Cancel
As Boolean)
    On Error Resume Next
    Set cb = CommandBars("StatPopup")
    If Err.Number <> 0 Then CreatePopup '创建快捷菜单
    CommandBars("StatPopup").ShowPopup '显示快捷菜单
    Cancel = True                      '禁止显示内置快捷菜单
End Sub
```

在以上代码中，通过将快捷菜单 StatPopup 赋值给一个对象变量的操作，判断该快捷菜单是否存在。若该快捷菜单不存在，将产生错误。通过 On Error 语句捕获错误，调用 CreatePopup 子过程创建菜单，然后使用 ShowPopup 方法将快捷菜单显示出来。

最后将 Cancel 参数设置为 True，确保不显示正常情况下的快捷菜单。

21.4.3　添加菜单项到内置快捷菜单

在上节的代码中，开发人员新建一个快捷菜单，在合适的区域取代系统内置快捷菜单。在更多的情况下，只需要在已有内置快捷菜单中添加一些菜单项，可达到增加新功能的同时，也能使用系统原有功能的目的。

如图 21-12 所示，在单元格、列标签和行标签上分别单击鼠标右键，可看到分别新增了对应的快捷菜单项。

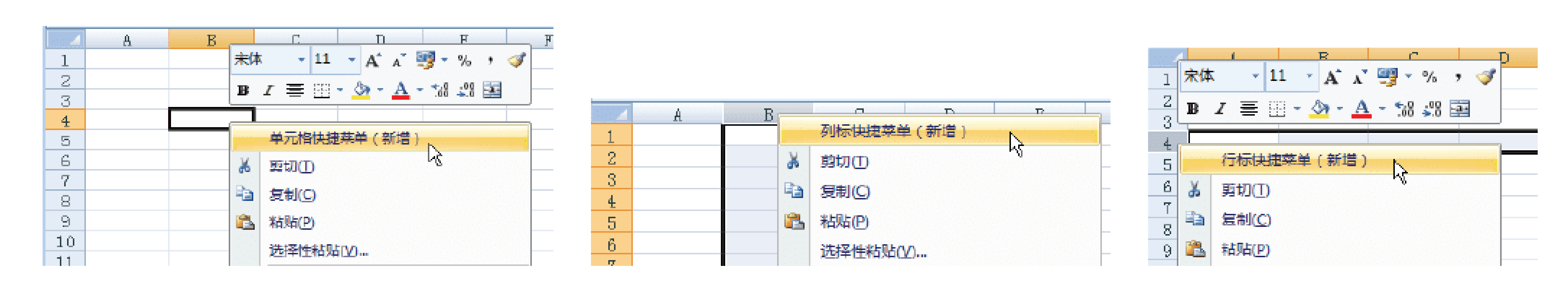


图 21-12　添加菜单项

要在已有快捷菜单中添加菜单项，首先需要获取对内置快捷菜单的引用。各快捷菜单的名称可从图 21-10 所示工作表中查得。例如，单元格快捷菜单的名称为 Cell，列标签快捷菜单的名称为 Column，行标签快捷菜单的名称为 Row。

若要向这些快捷菜单中添加菜单项，首先需获取对应菜单的引用，再使用以下代码添加菜单项：

```
Application.CommandBars("Cell").Controls.Add
```

下面列出具体的步骤。

（1）在工作簿的 Open 事件中编写以下 VBA 代码：

```
Private Sub Workbook_Open()
    Dim cbcs As CommandBarControls, cbc As CommandBarControl
    Set cbcs = Application.CommandBars("Cell").Controls
    Set cbc = cbcs.Add(Type:=msoControlButton, before:=1, temporary:=True)
    With cbc
        .Caption = "单元格快捷菜单（新增）"
        .OnAction = "cellmenu1_sub"
        .Tag = "cellmenu1"
    End With
    Set cbcs = Application.CommandBars("Column").Controls
    Set cbc = cbcs.Add(Type:=msoControlButton, before:=1, temporary:=True)
    With cbc
        .Caption = "列标签快捷菜单（新增）"
        .OnAction = "colmenu1_sub"
        .Tag = "colmenu1"
    End With
    Set cbcs = Application.CommandBars("Row").Controls
    Set cbc = cbcs.Add(Type:=msoControlButton, before:=1, temporary:=True)
    With cbc
        .Caption = "行标签快捷菜单（新增）"
        .OnAction = "rowmenu1_sub"
        .Tag = "rowmenu1"
    End With
End Sub
```

以上代码首先获取单元格的快捷菜单，接着使用 Add 方法向单元格快捷菜单中添加自定义菜单项。行标签和列标签快捷菜单的代码依此类似。

（2）关闭工作簿之前，删除上步新增的菜单项，具体代码如下：

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)   '关闭工作簿前删除新增
                                                       的快捷菜单
    Dim cbc As CommandBarControl
    For Each cbc In Application.CommandBars("Cell").Controls
        If cbc.Tag = "cellmenu1" Then cbc.Delete
    Next
    For Each cbc In Application.CommandBars("Column").Controls
        If cbc.Tag = "colmenu1" Then cbc.Delete
    Next
    For Each cbc In Application.CommandBars("Row").Controls
        If cbc.Tag = "rowmenu1" Then cbc.Delete
    Next
End Sub
```

（3）在 VBE 中插入一个模块，分别编写选择菜单项时调用的子过程。本例只是显示一

个提示信息对话框，实际系统中应将选择菜单要执行的代码放在此处。

```
Sub cellmenu1_sub()
    MsgBox "调用新增加的单元格快捷菜单！", vbInformation + vbOKOnly, "快捷菜单"
End Sub
Sub colmenu1_sub()
    MsgBox "调用新增加的列标签快捷菜单！", vbInformation + vbOKOnly, "快捷菜单"
End Sub
Sub rowmenu1_sub()
    MsgBox "调用新增加的行标签快捷菜单！", vbInformation + vbOKOnly, "快捷菜单"
End Sub
```

编写好以上代码，关闭再重新打开工作簿，在工作表、行或列上单击鼠标右键时，在弹出的快捷菜单中将增加一个新的菜单项，如图 21-12 所示。

21.4.4　隐藏/禁止内置菜单项

对于系统内置的快捷菜单，通过 VBA 代码可隐藏指定的快捷菜单（或快捷菜单中的菜单项），也可禁止用户弹出某个快捷菜单（或禁止选择某个菜单项）。

所谓隐藏，是设置菜单项的 Visible 属性为 False 时，则该菜单项将不会显示在弹出的快捷菜单中，如果设置为 True，则又将显示出来。

如果要禁止某个快捷菜单（菜单项），则需要设置快捷菜单（或菜单项）的 Enabled 属性，设置为 True，表示允许用户右击弹出该快捷菜单（或单击选择菜单项），设置为 False，表示用户右击时将不弹出对应的快捷菜单（禁止用户选择该菜单项——显示为灰色）。

例如，在正常情况下，用户在工作表标签上单击右键将弹出操作工作表的快捷菜单，在该快捷菜单中包含了对工作表进行操作的各种命令，如图 21-13 所示。

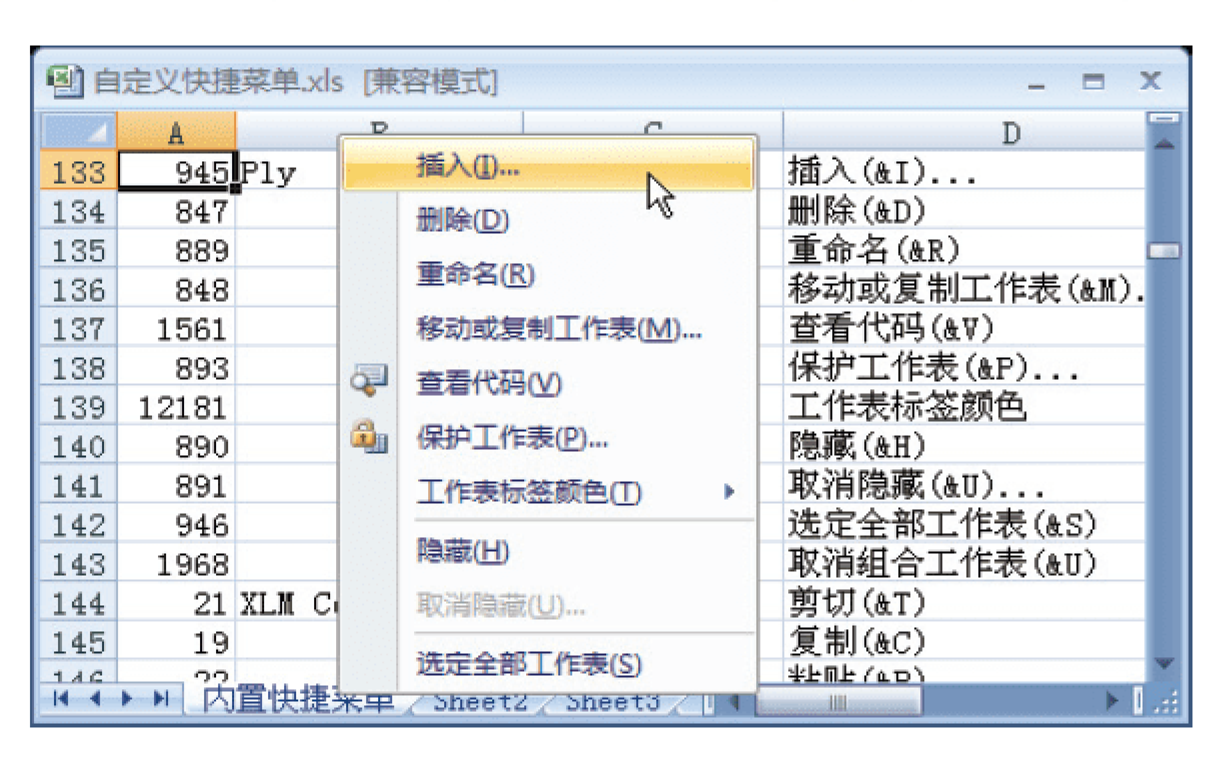


图 21-13　工作表快捷菜单

在某些情况下，可能需要禁止弹出如图 21-13 所示的快捷菜单，则可运行以下代码：

```
Sub 禁止快捷菜单()
    Application.CommandBars("ply").Enabled = False
End Sub
```

以上代码通过设置名为 ply 的快捷菜单的 Enabled 属性为 False，达到禁止显示该菜单的目的。

提示：在图 21-10 所示的工作表中列出了内置快捷菜单的名称、显示文本、ID 号，要控制其他快捷菜单，可查看相关信息。

运行以下代码，又可允许弹出名为 ply 的快捷菜单：

```
Sub 允许快捷菜单()
    Application.CommandBars("ply").Enabled = True
End Sub
```

对于快捷菜单中的菜单项，通过 FindControl 方法查找，然后可对找到的菜单项进行控制。例如，以下代码禁止使用工作表快捷菜单中的【插入】菜单项。执行以下代码后，工作表的快捷菜单如图 21-14 所示，【插入】菜单项为灰色，禁止用户选择。

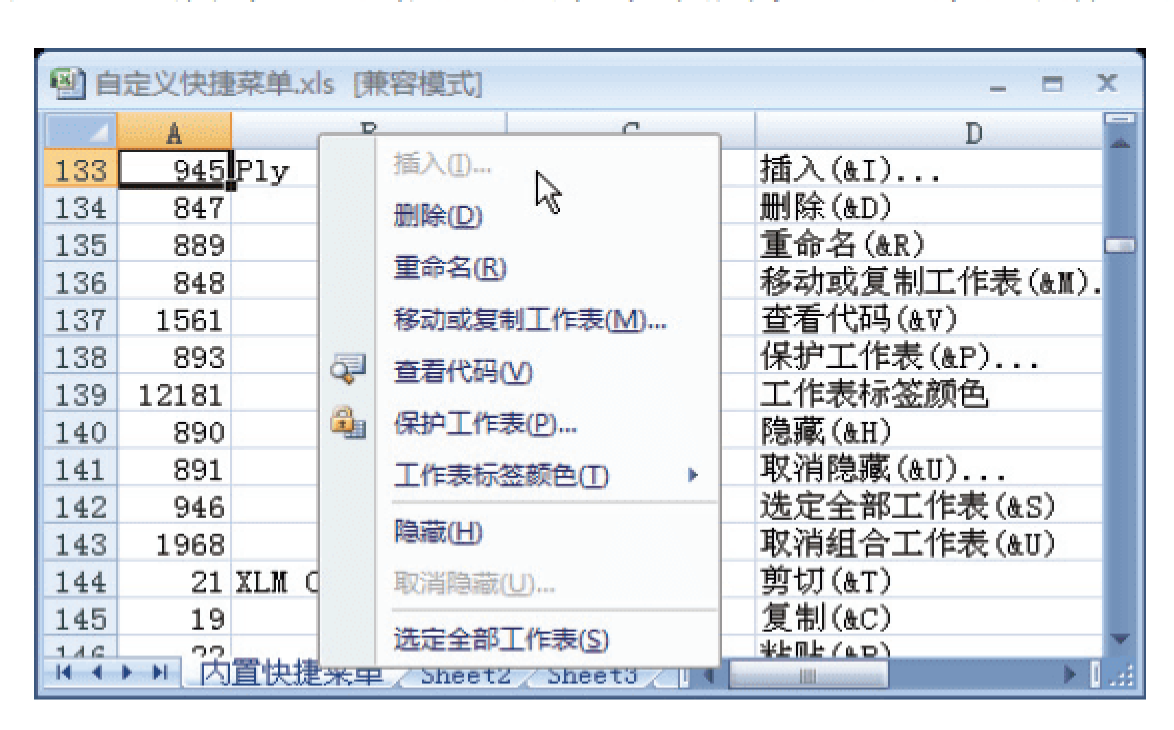


图 21-14　禁止【插入】菜单项

```
Sub 禁止插入菜单()
    Application.CommandBars("ply").FindControl(ID:=945).Enabled = False
End Sub
```

提示：执行以上代码后，Excel 将一直禁止用户使用【插入】菜单项，在合适的情况下，应设置允许用户使用。

同样，设置其 Enabeld 属性为 True，则允许用户使用。

```
Sub 允许插入菜单()
    Application.CommandBars("ply").FindControl(ID:=945).Enabled = True
End Sub
```

若要隐藏【插入】菜单项，则可执行以下代码：

```
Sub 隐藏插入菜单()
    Application.CommandBars("ply").FindControl(ID:=945).Visible = False
End Sub
```

执行以上代码，工作表快捷菜单如图 21-15 所示，【插入】菜单项被隐藏了。

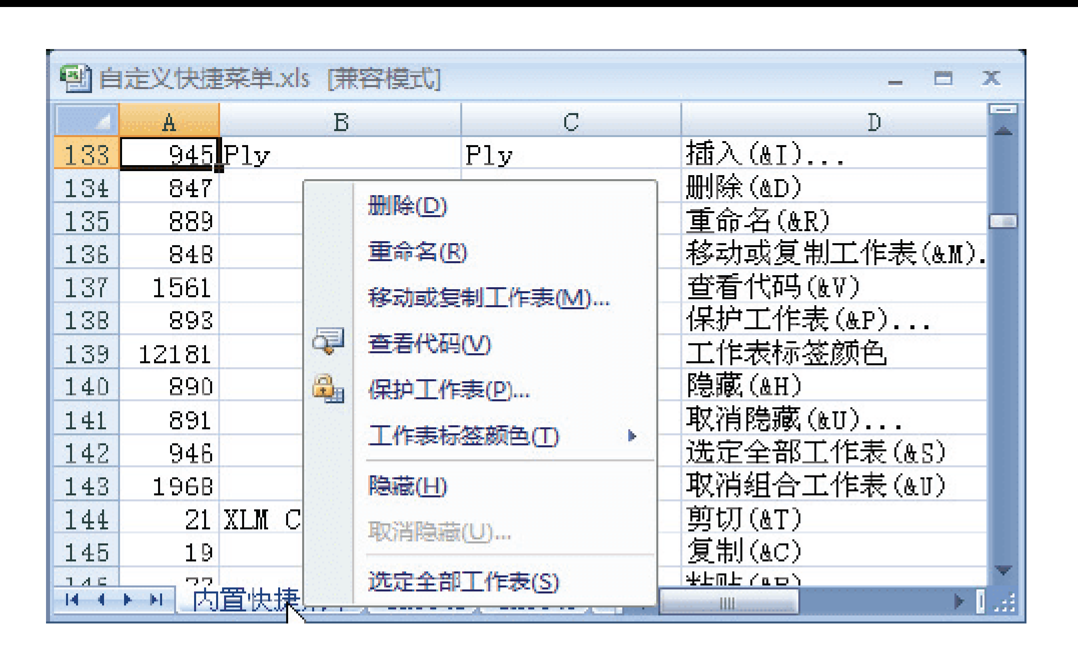


图 21-15　隐藏【插入】菜单项

使用以下代码可将【插入】菜单项显示出来：

```
Sub 显示插入菜单()
    Application.CommandBars("ply").FindControl(ID:=945).Visible = True
End Sub
```

21.5　自定义工具栏

在 Excel 2003 及以前版本中，提供了丰富的内置工具栏，方便用户对工作表中的数据进行操作。

在 Excel 2007 中，用功能区取代了这些工具栏。为了兼容早期版本，在 Excel 2007 中仍然可以用 VBA 代码访问这些内置工具栏，只是不能将其显示在工作界面中。使用 VBA 代码自定义的工具栏，将显示在功能区【加载项】选项卡的【自定义工具栏】组中。如果自定义了多个工具栏，将全部显示在该组中。

21.5.1　内置工具栏

在 Excel 2003 及以前版本中，工具栏显示如图 21-16 所示。



图 21-16　工具栏

一个工具栏为一个 CommandBar 对象，工具栏中包含多个按钮，每个按钮为一个 CommandBarButton 对象。

使用以下代码可在工作表“内置工具栏”中显示内置工具栏的名称，及每个工具栏所包含的按钮名称。

```
Sub 显示内置工具栏()
    Dim cb As CommandBar, cbc As CommandBarControl
    Dim i As Long, j As Long
```

```
    On Error Resume Next

    With Worksheets("内置工具栏")
        i = 2
        For Each cb In CommandBars                    '循环处理每个命令栏
            If cb.Type = msoBarTypeNormal And cb.BuiltIn = True Then
                .Cells(i, 2) = cb.Name                '获取命令栏名称
                .Cells(i, 3) = cb.NameLocal           '获取命令栏中文名称
                For Each cbc In cb.Controls           '循环处理每个命令栏中的控件按钮
                    .Cells(i, 1) = cbc.ID             '控件按钮 ID
                    .Cells(i, 4) = cbc.Caption        '控件按钮显示文字
                    i = i + 1
                Next
            End If
        Next
        .Cells.Columns.AutoFit
    End With
End Sub
```

运行以上代码，可得到如图 21-17 所示的内容。在“内置工具栏”工作表中列出了每个工具栏按钮的 ID、工具栏名称等信息。

自走义工具栏.xls [兼容模式]

	A	B	C	D
1	ID	工具栏名称	中文名称	控件
2	1031	WordArt	艺术字	艺术字(&W)...
3	2094			编辑文字(&X)...
4	1606			艺术字库(&W)
5	962			对象(&O)...
6	1058			艺术字形状(&W)
7	1063			艺术字字母高度相同(&W)
8	1061			艺术字竖排文字(&W)
9	1059			艺术字对齐方式(&W)
10	1060			艺术字字符间距(&W)
11	2619	Picture	图片	来自文件(&F)...
12	1403			颜色(&C)
13	1064			增加对比度(&M)
14	1065			降低对比度(&L)
15	1066			增加亮度(&M)
16	1067			降低亮度(&L)

内置工具栏 Sheet2 Sheet3

图 21-17　内置工具栏

21.5.2　创建工具栏

使用 VBA 代码创建工具栏与创建菜单栏的方法类似，首先向 CommandBars 集合中添加一个 CommandBar 对象，再向该对象中添加 CommandBarControl 对象（因为其类型为 msoControlButton，实质上添加的是一个 CommandBarButton 对象），再设置该对象的各参数即可。

例如，下面的代码可创建一个工具栏：

```
Sub CreateToolsBar()
    Dim mybar As CommandBar, mybutton1 As CommandBarButton
    删除自定义工具栏                                '调用子过程删除同名工具栏

    Set mybar = Application.CommandBars.Add(Name:="销售管理")    '增加工具栏
```

```
    With mybar
        .Visible = True                         '显示工具栏
        .Protection = msoBarNoCustomize         '禁止用户对工具栏进行添加或删除
    End With

    Set mybutton1 = mybar.Controls.Add(msoControlButton)
                                                '向工具栏中添加一个按钮
    With mybutton1
        .FaceId = 720                           '设置按钮的图标
        .Style = msoButtonIconAndCaption        '设置按钮的风格
        .Caption = "进货"                       '设置按钮的提示信息
        .OnAction = "供货单"                    '设置单击按钮时调用的子过程
        .Visible = True                         '显示按钮
    End With

    Set mybutton1 = mybar.Controls.Add(msoControlButton)
    With mybutton1
        .FaceId = 59
        .Style = msoButtonIconAndCaption
        .Caption = "销售"
        .OnAction = "销货单"
       Visible = True
    End With

    Set mybutton1 = mybar.Controls.Add(msoControlButton)
    With mybutton1
        .FaceId = 226
        .Style = msoButtonIconAndCaption
        .Caption = "存货统计"
        .OnAction = "统计"
        .Visible = True
    End With

    Set mybutton1 = mybar.Controls.Add(msoControlButton)
    With mybutton1
        .FaceId = 226
        .Style = msoButtonIconAndCaption
        .Caption = "进货报表"
        .OnAction = "进货报表"
        .Visible = True
    End With

    Set mybutton1 = mybar.Controls.Add(msoControlButton)
    With mybutton1
        .FaceId = 226
        .Style = msoButtonIconAndCaption
        .Caption = "销售报表"
        .OnAction = "销售报表"
        .Visible = True
    End With
End Sub
```

以上代码，首先调用名为“删除自定义工具栏”的过程删除系统中同名的工具栏，接着使用 Add 方法新建工具栏，再向新建的工具栏中添加按钮控件。

在向系统中添加自定义工具栏时，如果设置的名称与系统中已有命令栏的名称重复，程序将出现错误。因此在创建工具栏之前，应先判断是否有与创建工具栏名称相同的命令栏。若有该命令栏，则使用下面的子过程将其删除。

```
Sub 删除自定义工具栏()
    Dim cb As CommandBar
    For Each cb In CommandBars
        If cb.Name = "销售管理" Then
            cb.Delete
            Exit Sub
        End If
    Next
End Sub
```

提示：要完成实际的功能，还需要为每个按钮的 onAction 属性设置的子过程编写代码。

为了使用户进入系统就能使用工具栏，可以使用以下代码在工作簿的 Open 事件中调用创建工具栏的子过程：

```
Private Sub Workbook_Open()
    CreateToolsBar
End Sub
```

在关闭 Excel 应用程序时，应在工作簿的 BeforeClose 事件中编写以下代码删除自定义的工具栏。否则，在用户下次打开 Excel 进行编写其他工作簿时，工具栏仍会显示出来。

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    删除自定义工具栏
End Sub
```

在 Excel 2003 中执行 CreateToolsBar 子过程，将创建如图 21-18 所示的工具栏。

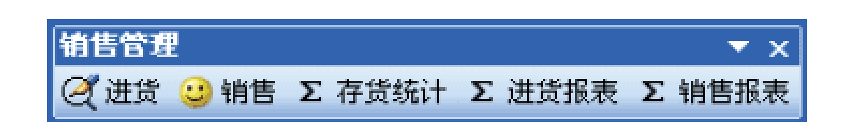


图 21-18　Excel 2003 中的自定义工具栏

在 Excel 2007 中执行 CreateToolsBar 子过程，创建的工具栏将不会浮于工作表上方，而放置在功能区【加载项】选项卡的【自定义工具栏】组中，如图 21-19 所示。



图 21-19　Excel 2007 中的自定义工具栏

第 5 部分　使用外部数据

在 Excel 2007 中，工作簿可保存海量数据。但是，在应用程序的过程中，可能还需要使用许多不是以 Excel 格式保存的数据。

Excel VBA 提供了访问外部数据的支持，通过 Excel 可访问其他程序，或其他程序创建的数据。Excel 在这方面具有很强的功能，可访问获取各种常见数据库中的数据、处理操作系统提供的文件系统、与 Office 其他应用程序组件共享数据、访问 Internet 中的数据等。

本部分共 4 章，分别介绍了 Excel 访问外部数据的方法。

第 22 章　控制其他 Office 程序

微软公司的 Office 套件包括了很多组件，如常用的 Word、Excel、PowerPoint、Outlook 等。这些组件既能单独使用，各组件之间也可相互调用。例如，使用 Excel 工作表中的数据生成 Word 文档。使用 VBA 可以方便地在各组件之间交换数据，本章将介绍具体的操作方法。

22.1　OLE 自动化技术简介

微软公司提供的 OLE 自动化技术，就是通过一个应用程序来控制另外一个应用程序的处理过程。在 Excel 环境中，通过自动化技术可控制其他 Office 应用程序。

22.1.1　OLE 简介

OLE/ActiveX/COM 技术是微软的核心应用技术。作为 COM 技术前身的 OLE，其最初含义是指在程序之间链接和嵌入对象数据（Object Link Embeded）。它提供了建立混合文档的手段，使得那些没有太多专业知识的用户能够很容易地协调多个应用程序完成混合文档的建立。OLE 技术包含以下功能。

- ❑ OLE 自动化：一个程序有计划地控制另一个程序的能力。
- ❑ OLE 控件：又称为 ActiveX 控件，是一个小型的组件程序，可嵌入到另外的程序，提供自己的专有功能。
- ❑ OLE 文档：又称为 ActiveX 文档，不仅支持简单链接和嵌入，还支持在位激活、拖放等功能。

Office 套件中的各组件都支持 OLE 自动化。通过 OLE 自动化技术，任何用 Word、PowerPoint、Outlook 等组件能够完成的工作，都可以通过使用 OLE 自动化在 Excel 组件中完成。这时，可将其他 Office 组件作为一个对象来使用，并通过 VBA 代码可访问这些组件提供的功能。这样，可为 Excel 应用程序整合其他 Office 组件提供方便快捷的方法。

22.1.2　引用服务程序

在使用 OLE 自动化技术前，先了解两个术语：客户程序和服务程序。

- ❑ 客户程序：就是申请并使用其他组件的程序，例如，在 Excel 中访问 Word 的相关功能，其中，Excel 就是客户程序。
- ❑ 服务程序：通过特定的接口，将自己完成的一些功能提供给调用的应用程序。例如，在 Excel 中访问 Word 的相关功能，其中，Word 就是服务程序。

1. 引用对象

Office 套件中的各组件都提供一个对象模型，在其他组件中通过 VBA 代码访问服务程序的对象模型中的相关对象，以得到服务程序提供的相应功能。在 VBA 代码中，必须先将服务程序的对象模型引用添加到当前工程中，才能访问服务程序提供的功能。例如，要在 Excel 中访问 Word 的相关功能，可通过以下步骤将 Word 对象模型引用到当前工程中。

（1）在 Excel 中按 Alt+F11 进入 VBE 编辑器。

（2）单击主菜单【工具】|【引用】命令，打开【引用】对话框。

（3）在【引用】对话框中选中 Microsoft Word 12.0 Object Library 复选框，如图 22-1 所示。

（4）单击【确定】按钮，Word 对象模型就添加到工程中。这样，在工程中使用 VBA 代码即可访问 Word 对象模型中的对象。例如，在模块中声明变量时，可将变量声明为 Word 对象，如图 22-2 所示。

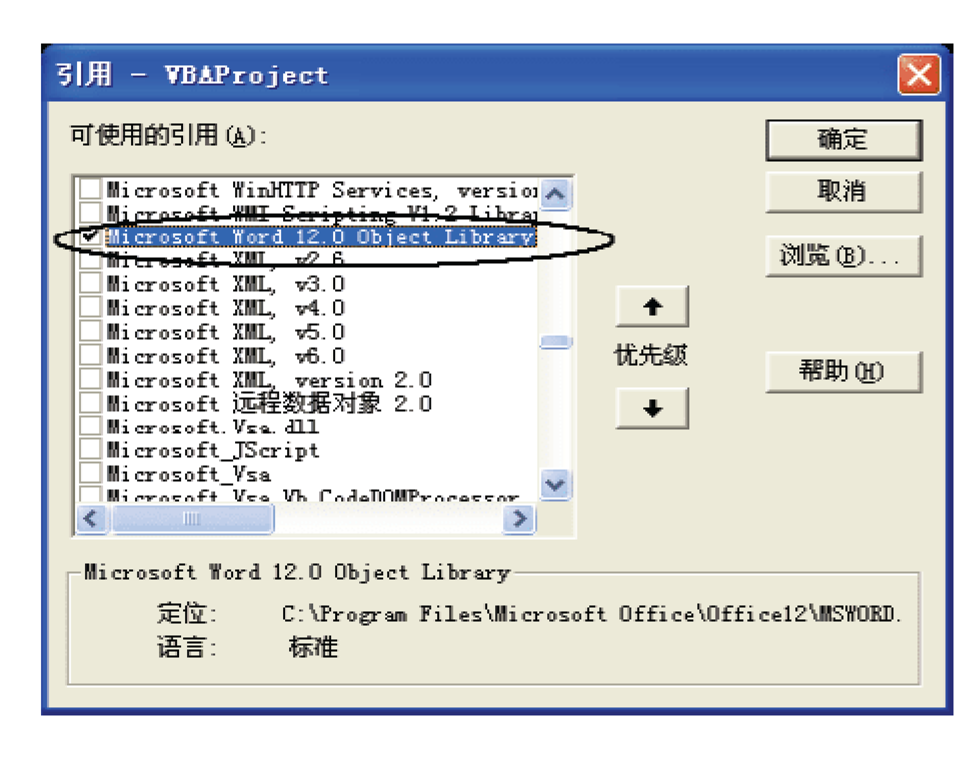


图 22-1　引用 Word 对象模型

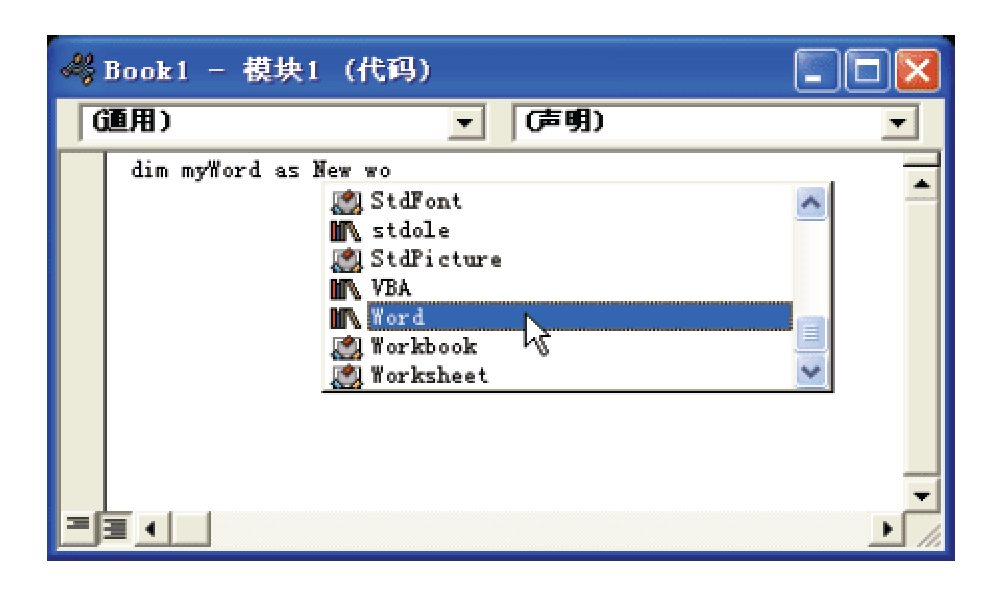


图 22-2　对象列表

2. 浏览对象的属性和方法

将 Word 对象库添加到当前工程后，可以通过【对象浏览器】窗口查看 Word 对象库的各种对象，以及这些对象的方法和属性。

（1）在 VBE 中按 F2 键打开【对象浏览器】窗口。

（2）单击左上角的【工程/库】下拉列表框，从中选择【Word】对象，如图 22-3 所示。

（3）在【搜索文字】下拉列表框中输入对象名称“paragraph”，单击【搜索】按钮，可在下方看到 Word 对象库中 paragraph（段落）对象的属性和方法，如图 22-4 所示。

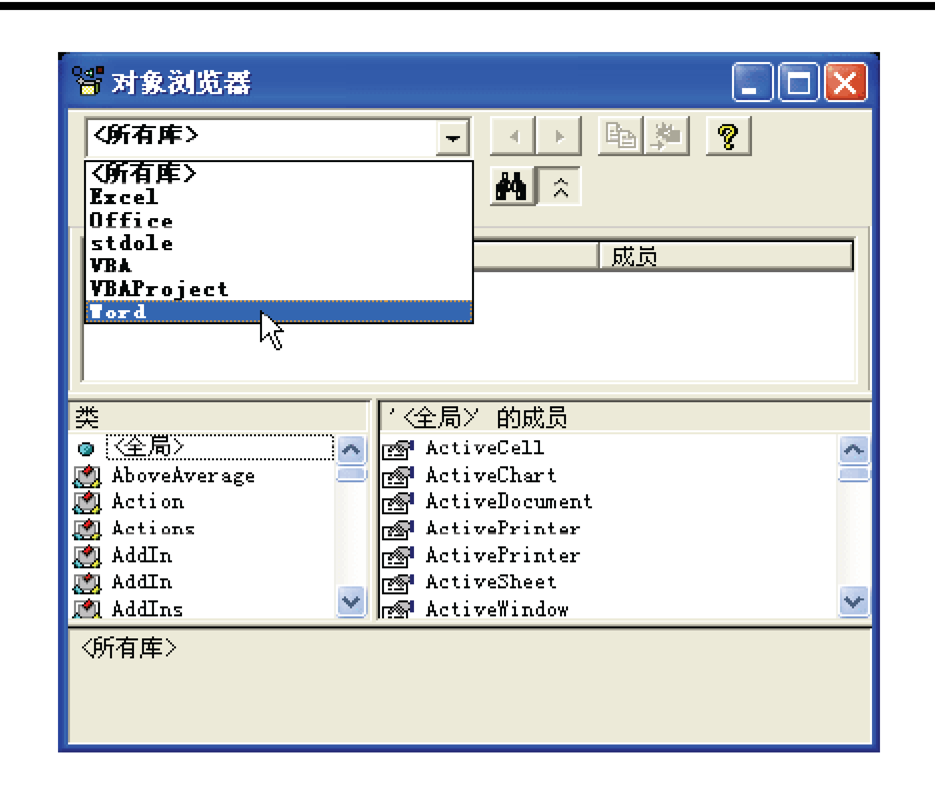


图 22-3　选择 Word 对象库

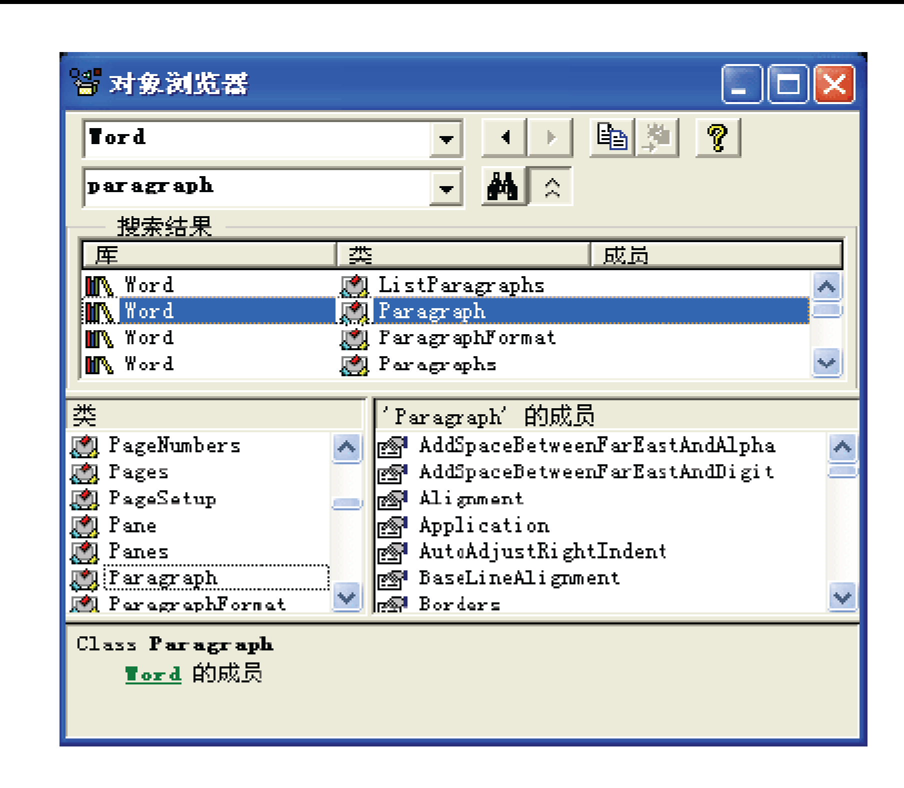


图 22-4　查看 paragraph 的属性和方法

22.1.3　实例化对象变量

将服务程序的对象模型添加到工程以后，还需要使用 VBA 代码声明一个对象变量，通过该对象变量来引用服务程序提供的功能。将服务程序的对象赋值给对象变量的过程，称为对象的实例化，可使用前期绑定和后期绑定两种方法来创建服务程序的实例。

1. 前期绑定

在定义对象变量时，若显式声明了变量的类型，该变量就只能存放该类型对象的引用。这种方式称为前期绑定。例如，以下代码定义对 Word 对象的引用：

```
Dim myWord As Word.Application
Set myWord=New Word.Application
```

以上代码中，首先定义了 myWord 变量为 Word.Application 对象，VBA 程序将自动检查该服务程序的可用性，如果没有检测到 Word 服务程序，程序将出现错误提示。通过前期绑定的方法，可以尽早地发现程序中的错误，方便程序调试。

使用前期绑定方式声明变量后，在代码中输入变量名加一个句点后，将显示属性/方法列表，如图 22-5 所示。



图 22-5　属性/方法列表

技巧： 在代码中若知道对象变量要使用的对象类型，最好使用前期绑定方式，以便及早发现程序中的错误，通过属性/方法列表可快速输入代码。

2．后期绑定

若将一个变量声明为 As Object 或 As Variant，VBA 在编译时将无法确定该变量引用哪种类型的对象。在程序运行时，根据赋值的对象类型来确定对象变量。这样，在运行时才能确定对象的属性和方法能否使用该变量。这种方式称为后期绑定。

例如，以下代码使用后期绑定方式来引用 Word 对象变量：

```
Dim myWord As Object
```

通过以上方式声明的变量，VBA 编译时还不知道要引用的对象类型。在程序运行过程中，可使用以下语句将 Word 对象类型绑定到 myWord 变量中：

```
Set  myWord=CreateObject("Word.Application")
```

使用以下语句可将变量 myWord 绑定为 PowerPoint 对象：

```
Set  myWord=CreateObject("PowerPoint.Application")
```

若使用后期绑定，则每次调用属性或方法时，VBA 都要检查属性方法的正确性。

由于没有具体限定变量 myWord 的类型，所以 VBA 不会检查 Word 服务程序是否存在。当 Word 应用程序不存在时，以上代码也可执行，只是将变量 myWord 赋值为空。只有在具体调用 Word 对象的属性或方法时，程序才会出错。这种方法不易检查程序错误。

提示： 用前期绑定还是后期绑定完全取决于声明变量的方式。对象的创建方式对此没有任何影响。

22.2　控制 Word 程序

Word 是一个非常普及的文字处理软件，很多资料都是使用 Word 格式保存的。在 Excel 中可打开 Word 文档、从 Word 文档中获取资料，还可通过 Excel 工作表中的数据创建生成 Word 文档。

22.2.1　了解 Word 对象模型

要在 Excel 中控制 Word，需首先使用本章 22.1 节中介绍的方法，将 Word 对象模型添加到工程中。

在 Word 中操作和改变的每一个东西都是一个对象，这些对象的相互关系组成了 Word 中的对象模型，如图 22-6 所示。

在 Word 中，文档、对话框、文本框、图形、图表甚至 Word 本身都是对象，同时，这些对象都有自己的属性和方法，因此，用户可通过编程来访问这些已有对象，改变它们

的属性，以完成某些较高级的功能。

下面简单介绍 Word 对象模型中的常用对象：

1．Application对象

Application 对象代表 Word 应用程序。Application 对象包含可返回最高级对象的属性和方法。例如，ActiveDocument 属性返回 Document 对象。

2．Bookmark对象

Bookmark 对象代表文档、选定内容或区域中的单个书签。Bookmark 对象是 Bookmarks 集合的成员。Bookmarks 集合包括【书签】对话框中列出的所有书签。

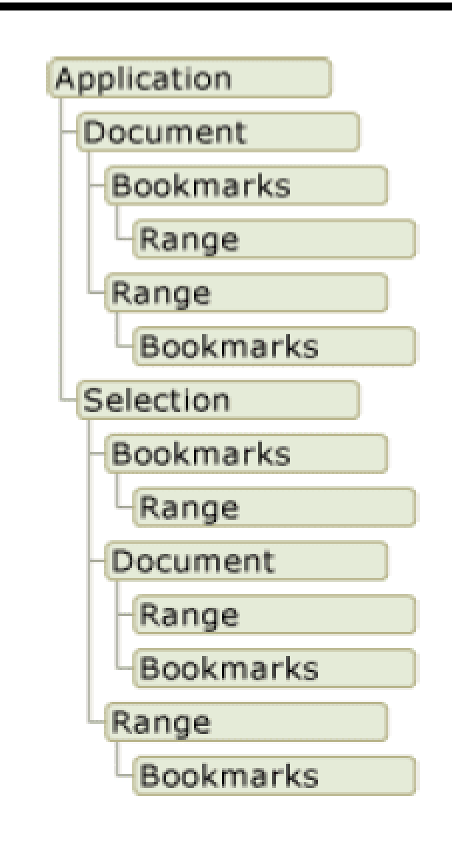


图 22-6　Word 对象模型

3．Document对象

Document 对象代表一个文档。Document 对象是 Documents 集合的成员。Documents 集合中包含当前在 Word 中打开的所有 Document 对象。

4．Paragraph对象

Paragraph 对象代表所选内容、范围或文档中的一个段落。Paragraph 对象是 Paragraphs 集合的一个成员。Paragraphs 集合包含所选内容、范围或文档中的所有段落。

5．Range对象

Range 对象代表文档中的一个连续区域。每个 Range 对象由一个起始字符位置和一个终止字符位置定义。

与书签在文档中的使用方法类似，Range 对象在 VBA 过程中用来标识文档的特定部分。但与书签不同的是，Range 对象只在定义该对象的过程运行时才存在，并且其独立于所选内容。也就是说，可以定义和处理一个范围而无需更改所选内容，还可以在文档中定义多个范围，但每个窗格中只能有一个所选内容。

6．Selection对象

Selection 对象代表窗口或窗格中的当前所选内容。所选内容代表文档中选定（或突出显示）的区域，如果文档中没有选定任何内容，则代表插入点。每个文档窗格只能有一个 Selection 对象，并且在整个应用程序中只能有一个活动的 Selection 对象。

可以使用 Selection 属性返回 Selection 对象。如果 Selection 属性未使用对象限定符，则 Word 返回活动文档窗口的活动窗格中的所选内容。

22.2.2　打开 Word 文档

在 Excel 中打开 Word 文档的操作比较简单，可以在 VBA 中创建对 Word 对象模型的引用，再使用 Documents 集合对象的 Open 方法即可打开指定的 Word 文档。Documents 集

合对象提供一个名为 Open 的方法，可用来打开文档，具体的语法格式如下：

```
Documents.Open(FileName, ConfirmConversions, ReadOnly, AddToRecentFiles,
PasswordDocument, PasswordTemplate, Revert, WritePasswordDocument,
WritePasswordTemplate, Format, Encoding, Visible, OpenConflictDocument,
OpenAndRepair,
DocumentDirection, NoEncodingDialog)
```

参数 FileName 为要打开的 Word 文档名（可包含路径），其余参数都可省略。

在 Excel 中打开 Word 文档的 VBA 代码如下：

```
Sub 打开 Word 文档()
    Dim sFName As String, strFilt As String, strTitle As String
    Dim docApp As Object

    strFilt = "Word 文档(*.doc;*.docx;*.docm),*.doc;*.docx;*.docm,"
    strTitle = "打开 Word 文档"

    sFName = Application.GetOpenFilename _
        (filefilter:=strFilt, _
        Title:=strTitle)
    If sFName = "False" Then Exit Sub

    Set docApp = CreateObject("Word.Application") '实例化 Word 对象变量
    docApp.Documents.Open sFName                  '打开 Word 文档
    docApp.Visible = True                         '显示 Word 应用程序

    Set docApp = Nothing                          '释放对象变量
End Sub
```

以上代码首先让用户选择要打开的 Word 文档，接着实例化 Word 对象变量，再调用 Word 对象模型中 Documents 集合的 Open 方法打开文档。

执行以上过程，将显示如图 22-7 所示的对话框。在对话框中列出了扩展名为.doc、.docx 和.docm 的 Word 文档。

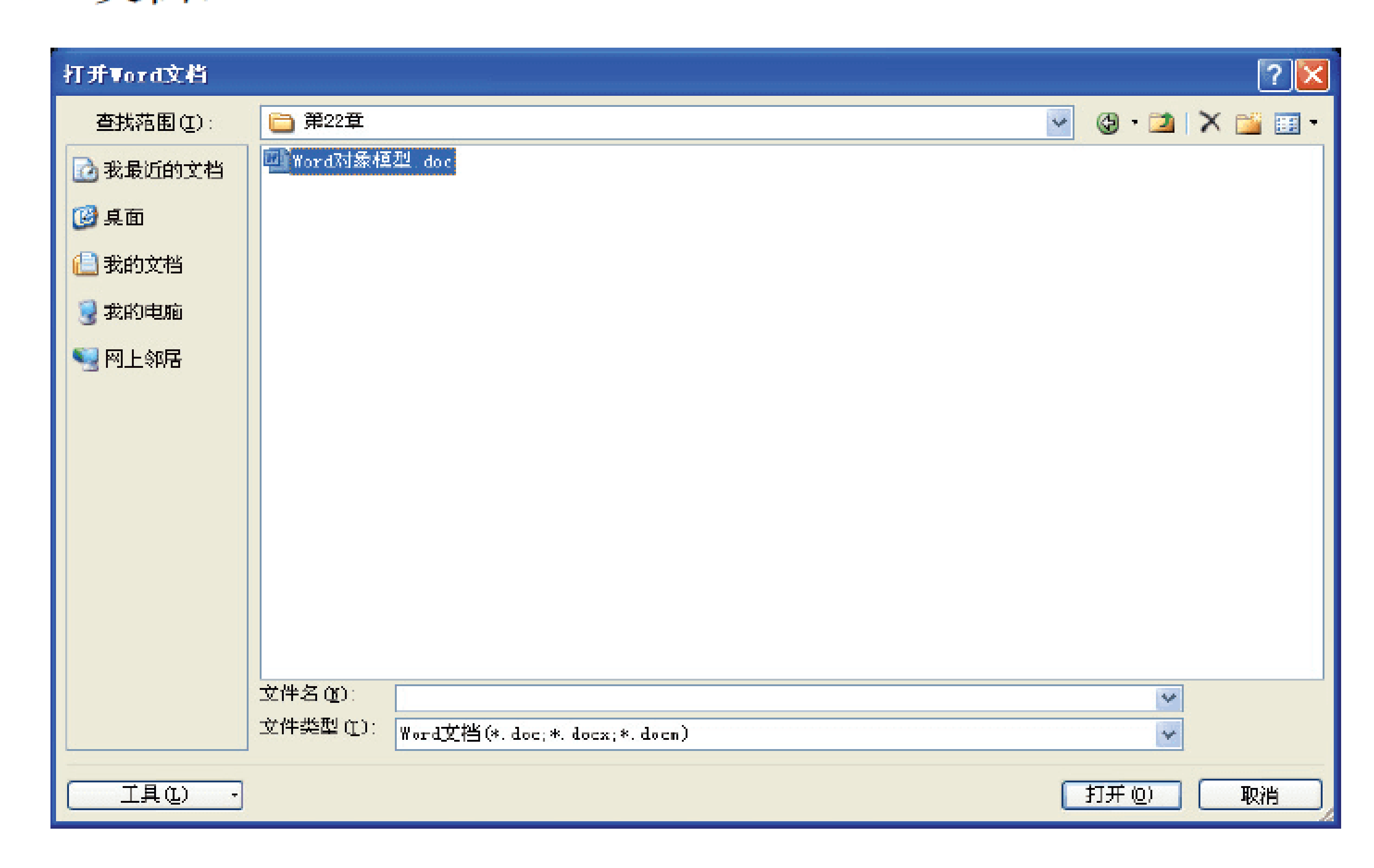


图 22-7 【打开 Word 文档】对话框

在如图 22-7 所示的对话框中选择文件“Word 对象模型.doc”，单击【打开】按钮即可启动 Word 程序打开选中的文档，如图 22-8 所示。

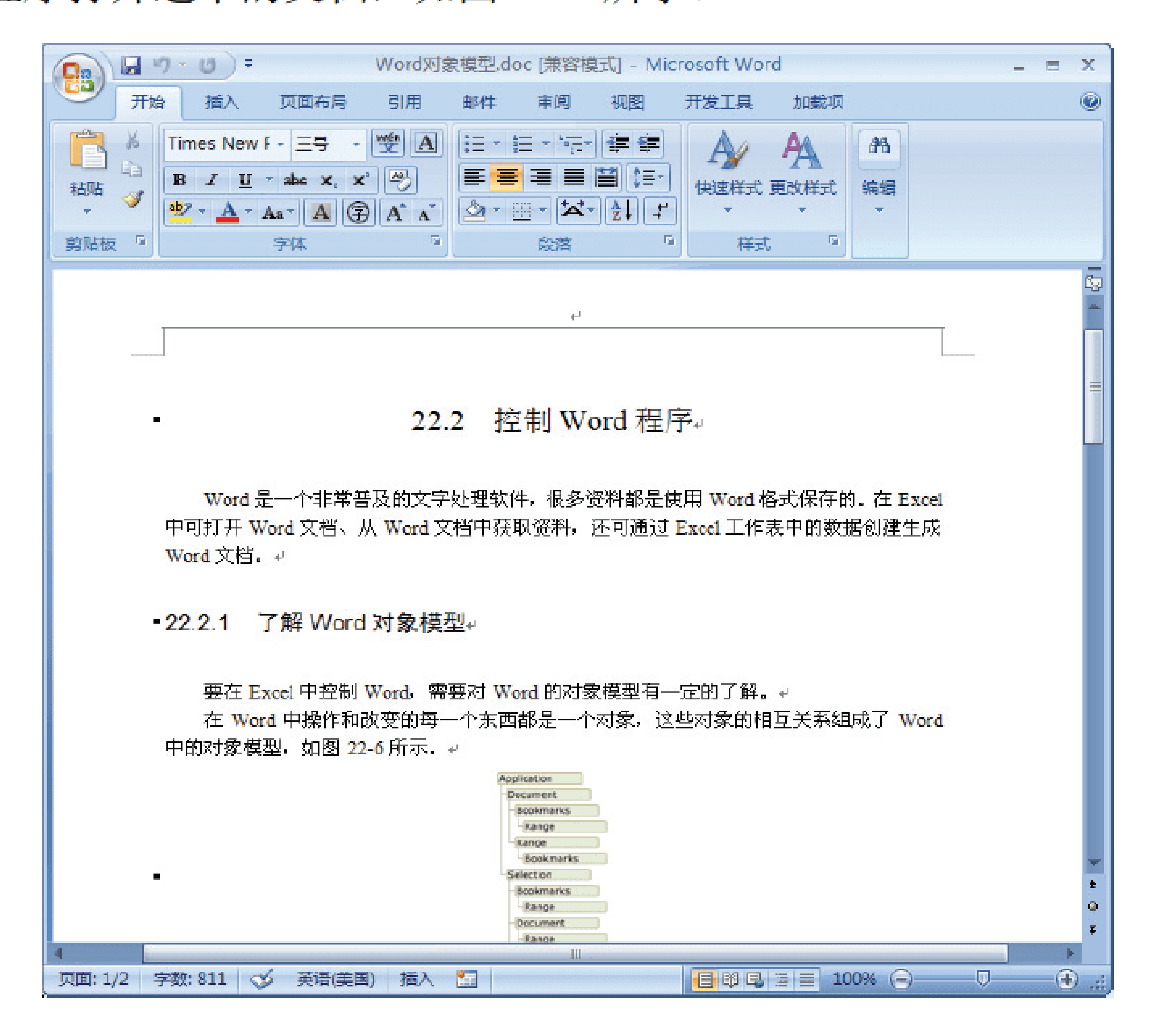


图 22-8 打开的 Word 文档

22.2.3 获取 Word 文档中的数据

在实际应用中，有时可能要从已有 Word 文档中获取数据，并填充到 Excel 工作表中。要从 Word 文档中读取数据，可通过 Paragraph 对象的 Range 属性，返回段落中包含的文档部分。

使用以下代码可以获取 Word 文档中的数据：

```
Sub 获取 Word 的数据()
    Dim sFName As String, strFilt As String, strTitle As String
    Dim docApp As Word.Application, pg As Word.Paragraph
    Dim i As Long, str1 As String

    strFilt = "Word 文档(*.doc;*.docx;*.docm),*.doc;*.docx;*.docm,"
    strTitle = "打开 Word 文档"
    sFName = Application.GetOpenFilename _
        (filefilter:=strFilt, _
        Title:=strTitle)
    If sFName = "False" Then Exit Sub

    Set docApp = CreateObject("Word.Application")  '实例化 Word 对象变量
    docApp.Documents.Open sFName                    '打开 Word 文档
    i = 1
```

```
    With docApp.ActiveDocument
        For Each pg In .Paragraphs                  '处理 Word 中的每一个段落
            str1 = pg.Range.Text                    '获取段落中的文本

            i = i + 1
            ActiveSheet.Cells(i, 1) = str1
        Next
    End With

    docApp.Quit                                     '退出 Word 文档
    Set docApp = Nothing                            '释放对象变量
End Sub
```

以上代码首先让用户选择要打开的 Word 文档，接着使用 Open 方法打开文档，然后通过 Paragraph 对象的 Range 属性获取段落文本，再将其赋值给 Excel 的单元格。

注意：以上代码没有设置 docApp 的 Visible 属性为 True，在程序运行过程中将看不到 Word 应用程序的窗口，结束程序前一定要使用 Word 对象的 Quit 方法退出隐藏的 Word 应用程序，否则该隐藏程序将一直在内存中。

执行以上代码，将首先显示【打开 Word 文档】对话框，选择要获取数据的 Word 文档“Word 对象模型.doc”，单击【打开】按钮，即可将该文档中的文本填入到工作表 Sheet2 中，如图 22-9 所示，并且各个段落的文字会填入 Excel 工作表的每一行中。

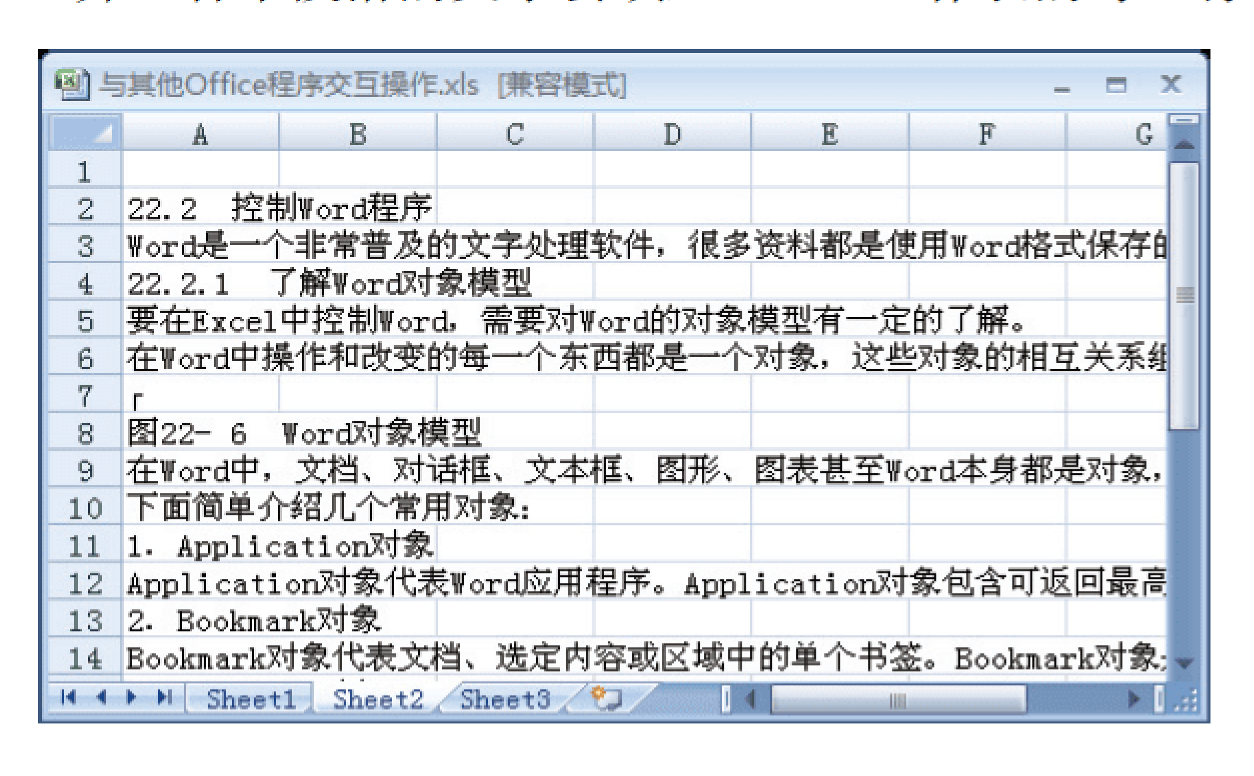


图 22-9　获取的 Word 文档数据

22.2.4　批量创建 Word 文档

Excel 的工作表可以保存成千上万条数据，通过 VBA 可使用这些数据创建格式相同而数据不同的 Word 文档。下面以示例演示用 Excel 工作表数据创建 Word 文档的方法。

汽车修理厂对维修出厂的车辆需打印《汽车维修竣工出厂合格证》，其格式如图 22-10 所示。在该合格证中，将车辆维修的数据填入打印即可。

为了提高效率，可将各维修车辆的数据输入到 Excel 工作表中保存，如图 22-11 所示。利用该工作表保存的数据可批量生成 Word 文档，再分别打印即可。

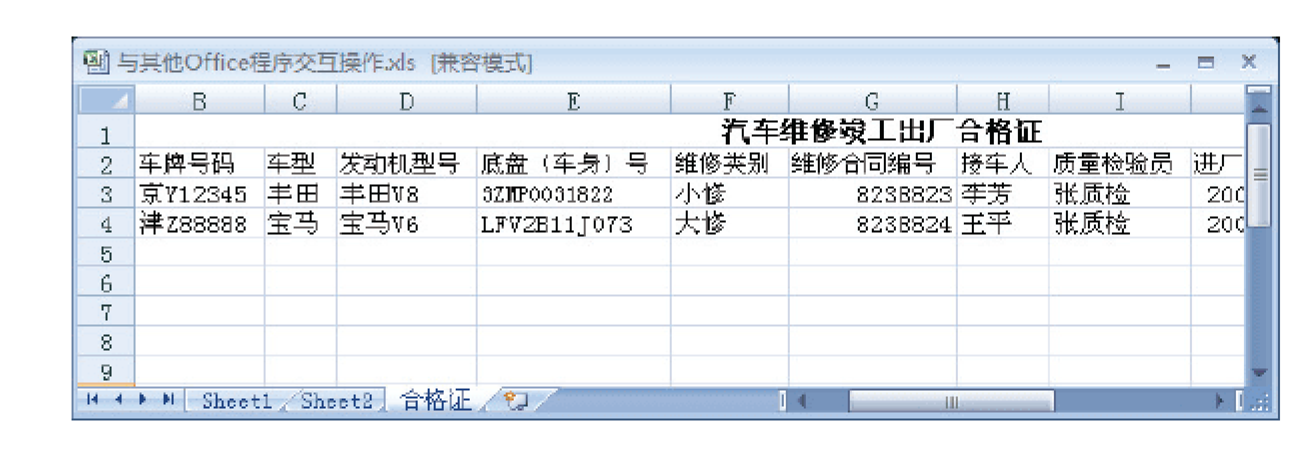
汽车维修竣工出厂合格证

托修方：天津分公司
车牌号码：津 Z88888
车型：宝马
发动机型号：宝马 V6
底盘（车身）号：LFV2B11J073
维修类别：大修

维修合同编号：8238824
接车人：王平
质量检验员：张质检

进厂日期：　2008 年 2 月 2 日
竣工日期：　2008 年 2 月 8 日
出厂日期：　2008 年 2 月 8 日
维修专用发票编号：335567

图 22-10　合格证



与其他Office程序交互操作.xls [兼容模式]

	B	C	D	E	F	G	H	I	
1					汽车维修竣工出厂合格证				
2	车牌号码	车型	发动机型号	底盘（车身）号	维修类别	维修合同编号	接车人	质量检验员	进厂
3	京Y12345	丰田	丰田V8	3ZMP0031822	小修	8238823	李芳	张质检	200
4	津Z88888	宝马	宝马V6	LFV2B11J073	大修	8238824	王平	张质检	200
5									
6									
7									
8									
9									

Sheet1 / Sheet2 / 合格证

图 22-11　Excel 工作表数据

要完成以上任务，可制作出一个合格证模板，在模板中插入多个书签，然后使用 VBA 程序读取 Excel 工作表中的数据，并分别填充到对应的书签中。

在 VBA 代码中使用 Selection 对象的多个方法来完成该工作，下面简单介绍这些方法的作用：

（1）GoTo 方法。

使用 GoTo 方法可将插入点移至紧靠指定项之前的字符位置，并返回一个 Range 对象，其语法格式如下：

```
表达式.GoTo(What, Which, Count, Name)
```

各参数的含义如下所述。

- What：范围或所选内容移动到的项目的种类。可设置为如表 22-1 所示的常量之一。
- Which：范围或所选内容要移动到的项目。可设置为如表 22-2 所示的常量之一。
- Count：文档中的项数。默认值为 1。只有正值有效。
- Name：如果 What 参数为 wdGoToBookmark、wdGoToComment、wdGoToField 或 wdGoToObject，则此参数指定一个名称。

表 22-1　移动类型

名　　称	值	描　　述	名　　称	值	描　　述
wdGoToBookmark	–1	书签	wdGoToLine	3	一个线段
wdGoToComment	6	批注	wdGoToObject	9	对象
wdGoToEndnote	5	尾注	wdGoToPage	1	页
wdGoToEquation	10	公式	wdGoToPercent	12	百分比
wdGoToField	7	域	wdGoToProofreadingError	15	校对错误
wdGoToFootnote	4	脚注	wdGoToSection	0	一节
wdGoToGrammaticalError	14	语法错误	wdGoToSpellingError	13	拼写错误
wdGoToGraphic	8	图形	wdGoToTable	2	一个表格
wdGoToHeading	11	标题			

表 22-2　移动位置

名　　称	值	描　　述	名　　称	值	描　　述
wdGoToAbsolute	1	绝对位置	wdGoToNext	2	所指定对象的下一个实例
wdGoToFirst	1	所指定对象的第一个实例	wdGoToPrevious	3	所指定对象的上一个实例
wdGoToLast	–1	所指定对象的最后一个实例	wdGoToRelative	2	相对于当前位置的位置

（2）TypeText 方法。

使用 Selection 对象的 TypeText 方法可插入指定的文本。

如果 ReplaceSelection 属性为 True，则表示用指定文本替换选定内容。如果 Replace Selection 属性为 False，则可在选定内容之前插入指定的文本。

完成批量生成《汽车维修竣工出厂合格证》Word 文档的具体步骤如下：

（1）在 Word 2007 中录入“汽车维修竣工出厂合格证”，并进行相应的版式设置，如图 22-12 所示。

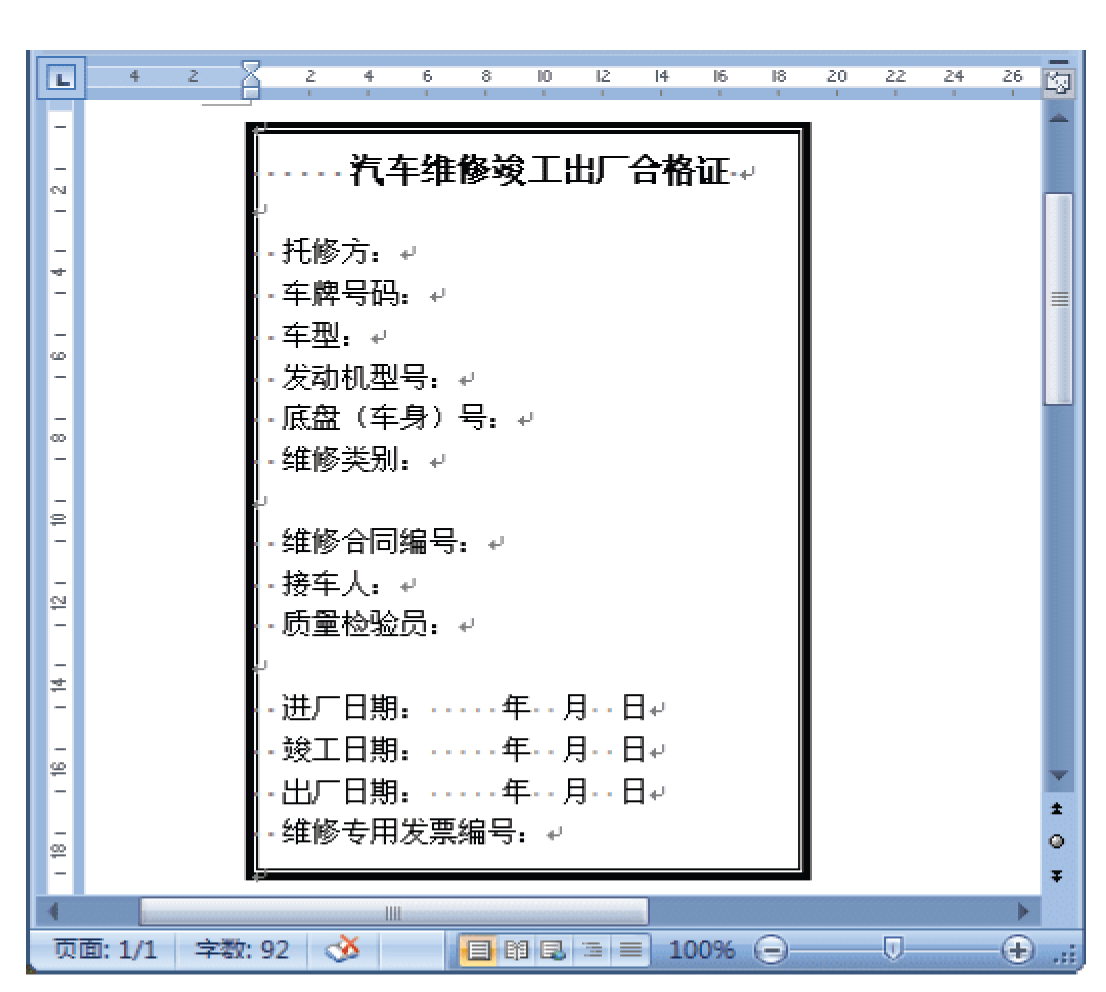


图 22-12　制作 Word 模板

（2）将光标定位在“托修方：”后面，在【插入】选项卡的【链接】组中，单击【书签】按钮，如图 22-13 所示。



图 22-13　插入书签

（3）在打开的【书签】对话框中输入书签名“托修方”，然后单击【添加】按钮，光

标处将添加一个书签，如图 22-14 左图所示。

（4）用同样的方法，在“车牌号码：”等数据项右侧分别插入书签，得到如图 22-14 右图所示的【书签】对话框。

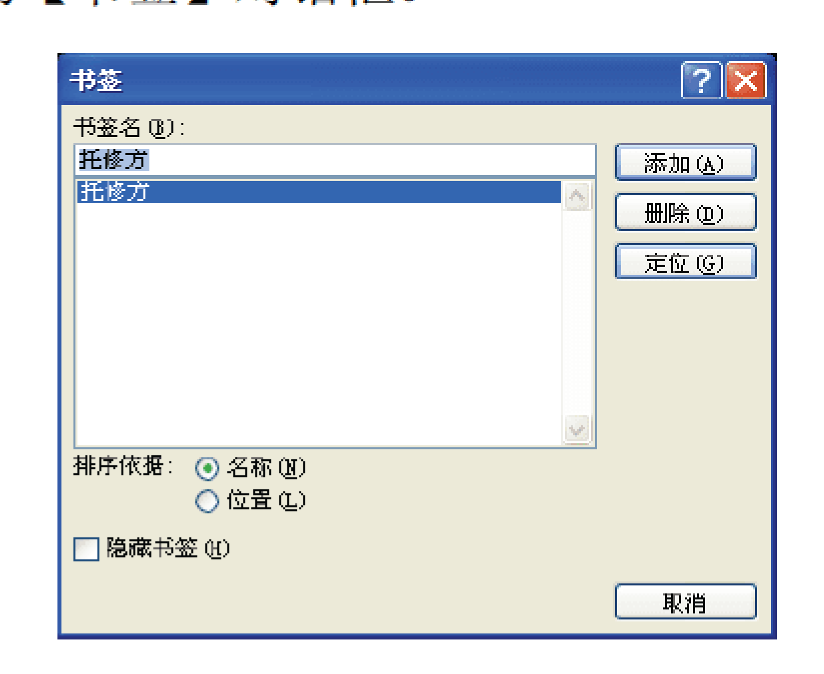




图 22-14 【书签】对话框

（5）将制作好的文档保存为“汽车维修竣工出厂合格证.dot”模板文件。

（6）在 Excel 中打开保存合格证相关资料的工作簿，按 Alt+F11 切换到 VBE 环境。

（7）在模块中编写“生成合格证”子过程，具体代码如下：

```
Sub 生成合格证()
    Dim i As Integer, r As Integer

    r = Sheets("合格证").[A65536].End(xlUp).Row
    For i = 3 To r                                  '处理每一行数据
        If Not CreateWord(i) Then Exit For          '调用 CreateWord 函数生成合格证
    Next
End Sub
```

以上代码首先从工作表“合格证”中获取数据行数，然后循环调用 CreateWord 函数生成指定行的合格证文档。

（8）在模块中编写子过程 CreateWord，用来生成 Word 文档。

```
Function CreateWord(ByVal r As Long) As Boolean    '创建 Word 文档
    Dim myWord As Word.Application, myDoc As Word.Document
    Dim str1 As String, dTemp As Date, sCarID As String
    Set myWord = New Word.Application
    sCarID = Worksheets("合格证").Cells(r, 2)
    With myWord
        Set myDoc = .Documents.Add(Template:=ThisWorkbook.Path & _
            "\汽车维修竣工出厂合格证 1.dot", Visible:=True)
                                                    '根据模板创建 Word 文档
        With .Selection
            .Goto What:=wdGoToBookmark, Name:="托修方" '查找书签
            .TypeText Text:=Worksheets("合格证").Cells(r, 1)
                                                    '在书签处输入文字
            .Goto What:=wdGoToBookmark, Name:="车牌号码"
            .TypeText Text:=sCarID
            .Goto What:=wdGoToBookmark, Name:="车型"
```

```
            .TypeText Text:=Worksheets("合格证").Cells(r, 3)
            .Goto What:=wdGoToBookmark, Name:="发动机型号"
            .TypeText Text:=Worksheets("合格证").Cells(r, 4)
            .Goto What:=wdGoToBookmark, Name:="底盘号"
            .TypeText Text:=Worksheets("合格证").Cells(r, 5)
            .Goto What:=wdGoToBookmark, Name:="维修类别"
            .TypeText Text:=Worksheets("合格证").Cells(r, 6)
            .Goto What:=wdGoToBookmark, Name:="维修合同编号"
            .TypeText Text:=Worksheets("合格证").Cells(r, 7)
            .Goto What:=wdGoToBookmark, Name:="接车人"
            .TypeText Text:=Worksheets("合格证").Cells(r, 8)
            .Goto What:=wdGoToBookmark, Name:="质量检验员"
            .TypeText Text:=Worksheets("合格证").Cells(r, 9)
            dTemp = Worksheets("合格证").Cells(r, 10)
            .Goto What:=wdGoToBookmark, Name:="进厂日期年"
            .TypeText Text:=DatePart("yyyy", dTemp)
            .Goto What:=wdGoToBookmark, Name:="进厂日期月"
            .TypeText Text:=DatePart("m", dTemp)
            .Goto What:=wdGoToBookmark, Name:="进厂日期日"
            .TypeText Text:=DatePart("d", dTemp)
            dTemp = Worksheets("合格证").Cells(r, 11)
            .Goto What:=wdGoToBookmark, Name:="竣工日期年"
            .TypeText Text:=DatePart("yyyy", dTemp)
            .Goto What:=wdGoToBookmark, Name:="竣工日期月"
            .TypeText Text:=DatePart("m", dTemp)
            .Goto What:=wdGoToBookmark, Name:="竣工日期日"
            .TypeText Text:=DatePart("d", dTemp)
            dTemp = Worksheets("合格证").Cells(r, 12)
            .Goto What:=wdGoToBookmark, Name:="出厂日期年"
            .TypeText Text:=DatePart("yyyy", dTemp)
            .Goto What:=wdGoToBookmark, Name:="出厂日期月"
            .TypeText Text:=DatePart("m", dTemp)
            .Goto What:=wdGoToBookmark, Name:="出厂日期日"
            .TypeText Text:=DatePart("d", dTemp)
            .Goto What:=wdGoToBookmark, Name:="维修专用发票编号"
            .TypeText Text:=Worksheets("合格证").Cells(r, 13)
        End With
        myDoc.SaveAs ThisWorkbook.Path & "\" & sCarID & ".doc", wdFormat
        Document
        myDoc.Close
        Set myDoc = Nothing '释放对象引用
    End With
    Set myWord = Nothing
    str1 = "车号: " & sCarID & " 的合格证生成完毕! " & _
        "是否生成下一辆车的合格证? "
    If MsgBox(str1, vbInformation + vbYesNo, "提示") = vbYes Then
        CreateWord = True   '函数返回值
    Else
        CreateWord = False
    End If
End Function
```

以上代码很长，但后面部分都是重复同一个操作。首先通过模板新建一个 Word 文档，接着使用 Goto 方法查找模板中的书签，再将数据表中的数据插入到书签处，最后保存生成的 Word 文档。

执行“生成合格证”子过程，即可循环生成工作表中每辆车的合格证，如图 22-10 所示。

22.3 控制 PowerPoint 程序

PowerPoint 主要用于设计制作广告宣传、产品演示的电子版幻灯片，制作的演示文稿可以通过计算机屏幕或者投影机播放。在 Excel 中，使用 VBA 可打开幻灯片、创建幻灯片。

22.3.1 了解 PowerPoint 对象模型

与操作 Word 对象类似，在 Excel 中操作 PowerPoint 对象也需要通过【引用】对话框来引用 PowerPoint 对象模型，如图 22-15 所示。

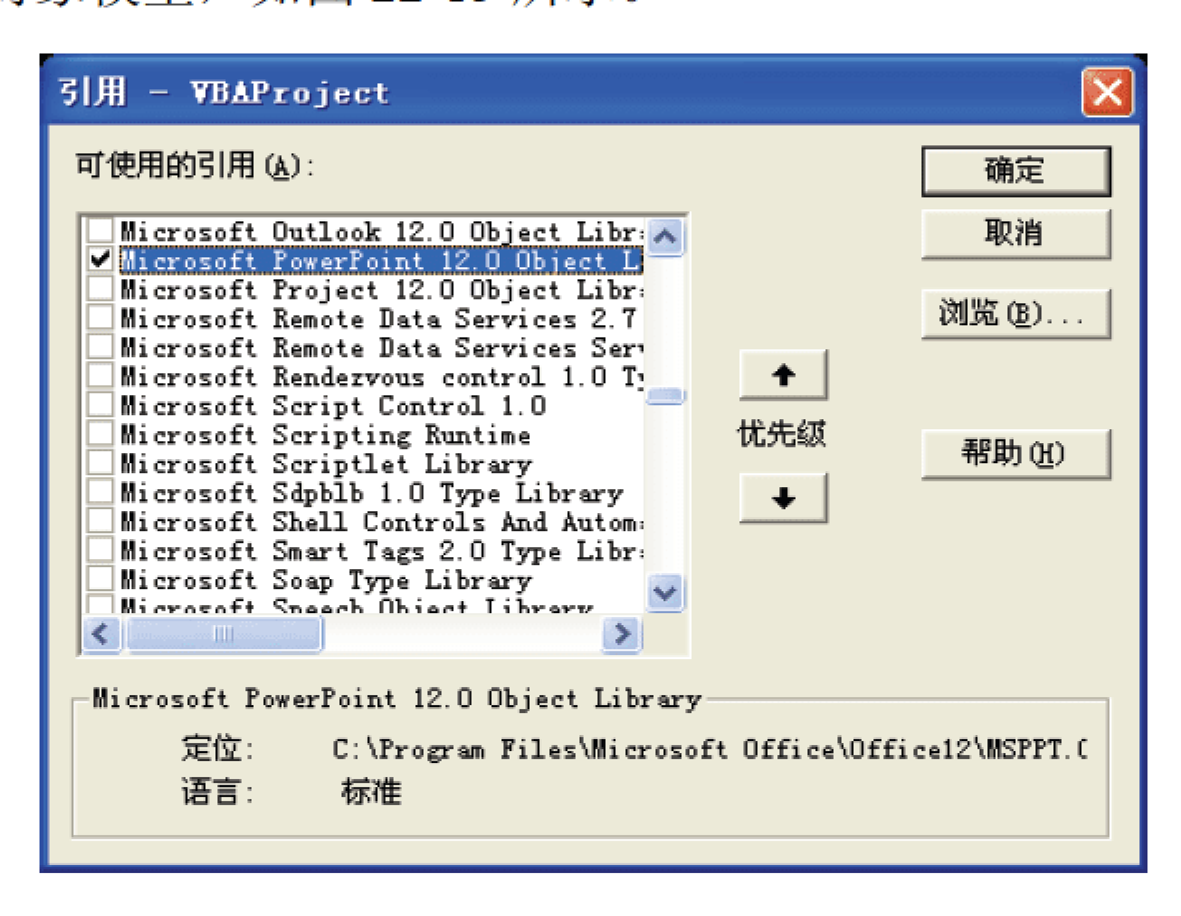


图 22-15 引用 PowerPoint 对象模型

PowerPoint 对象库中的常用对象如下：

1．Application对象

Application 对象代表整个 PowerPoint 应用程序。使用 PowerPoint 对象的 Application 属性可返回 Application 对象。

2．Presentation对象

Presentation 对象代表一个 PowerPoint 演示文稿。Presentation 对象属于 Presentations 集合中的成员。Presentations 集合中包含所有的 Presentation 对象，它们分别代表 PowerPoint

中所有打开的演示文稿。

使用 Presentations(index)可返回一个 Presentation 对象，其中，index 表示演示文稿的名称或索引号。演示文稿的名称就是其文件名，是否带文件扩展名均可，但不包含路径。

3．Slide对象

Slide 对象代表一个幻灯片。Slides 集合包含演示文稿中的所有 Slide 对象。

使用 Slides(index)（其中 index 为幻灯片名称或索引号）或 Slides.FindBySlideID（index）（其中 index 为幻灯片标识符）返回单个 Slide 对象。

通过 Slide 对象的属性和方法可控制单张幻灯片。

4．Slides集合

Slides 集合是指定演示文稿中所有 Slide 对象的集合。使用 Slides 属性返回 Slides 集合。使用 Add 方法创建新幻灯片并添加到集合。

22.3.2　打开演示文稿

与打开 Word 文档类似，在 Excel 中打开 PowerPoint 文档的操作比较简单。使用 Presentations 集合对象的 Open 方法即可打开指定的演示文稿，并且返回一个 Presentation 对象，该对象代表已打开的演示文稿，具体的语法格式如下：

```
表达式.Open(FileName, ReadOnly, Untitled, WithWindow, OpenConflictDocument)
```

参数 FileName 为要打开的文件名，其余参数都可省略。

下面的代码使用 Presentations 集合的 Open 方法打开演示文稿。

```
Sub 打开演示文稿()
    Dim sFName As String, strFilt As String, strTitle As String
    Dim pptApp As PowerPoint.Application          '声明 PPT 应用程序对象
    Dim pptPrs As PowerPoint.Presentation         '声明演示文稿对象

    strFilt = "PowerPoint 文件(*.ppt),*.ppt"
    strTitle = "打开演示文稿"

    sFName = Application.GetOpenFilename _
        (filefilter:=strFilt, _
        Title:=strTitle)
    If sFName = "False" Then Exit Sub

    Set pptApp = New PowerPoint.Application       '实例化 PowerPoint 应用程序
    pptApp.Visible = True
    Set pptPrs = pptApp.Presentations.Open(sFName) '打开演示文稿

    Set pptPrs = Nothing
    Set pptApp = Nothing
End Sub
```

以上代码首先让用户选择 PowerPoint 文件，再使用 Presentations 对象的 Open 方法打开演示文稿。

执行以上过程，将首先显示如图 22-16 所示的【打开演示文稿】对话框。

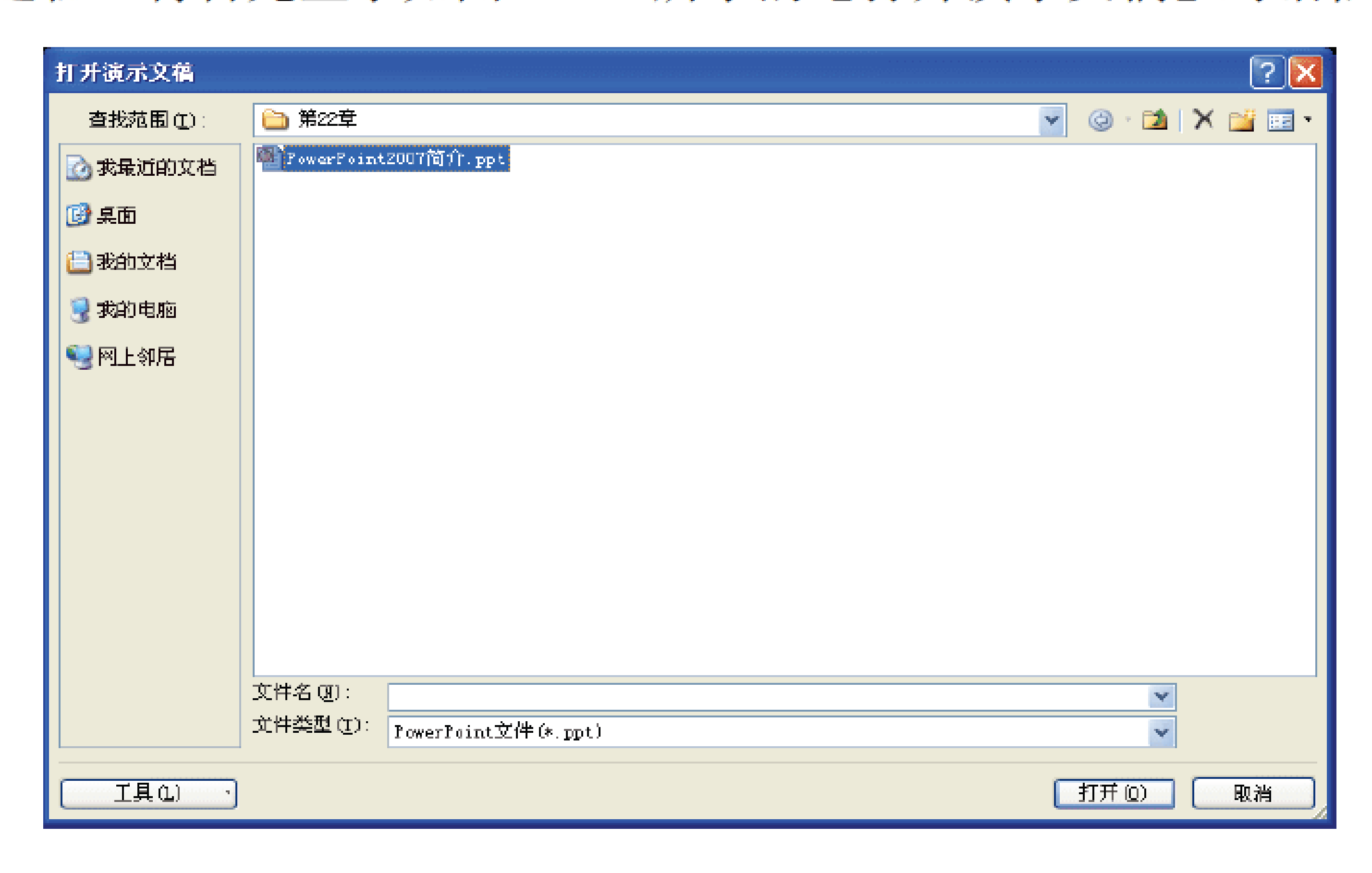


图 22-16 【打开演示文稿】对话框

在对话框中选择 PowerPoint2007 简介.ppt，单击【打开】按钮即可打开 PowerPoint 演示文稿，如图 22-17 所示。

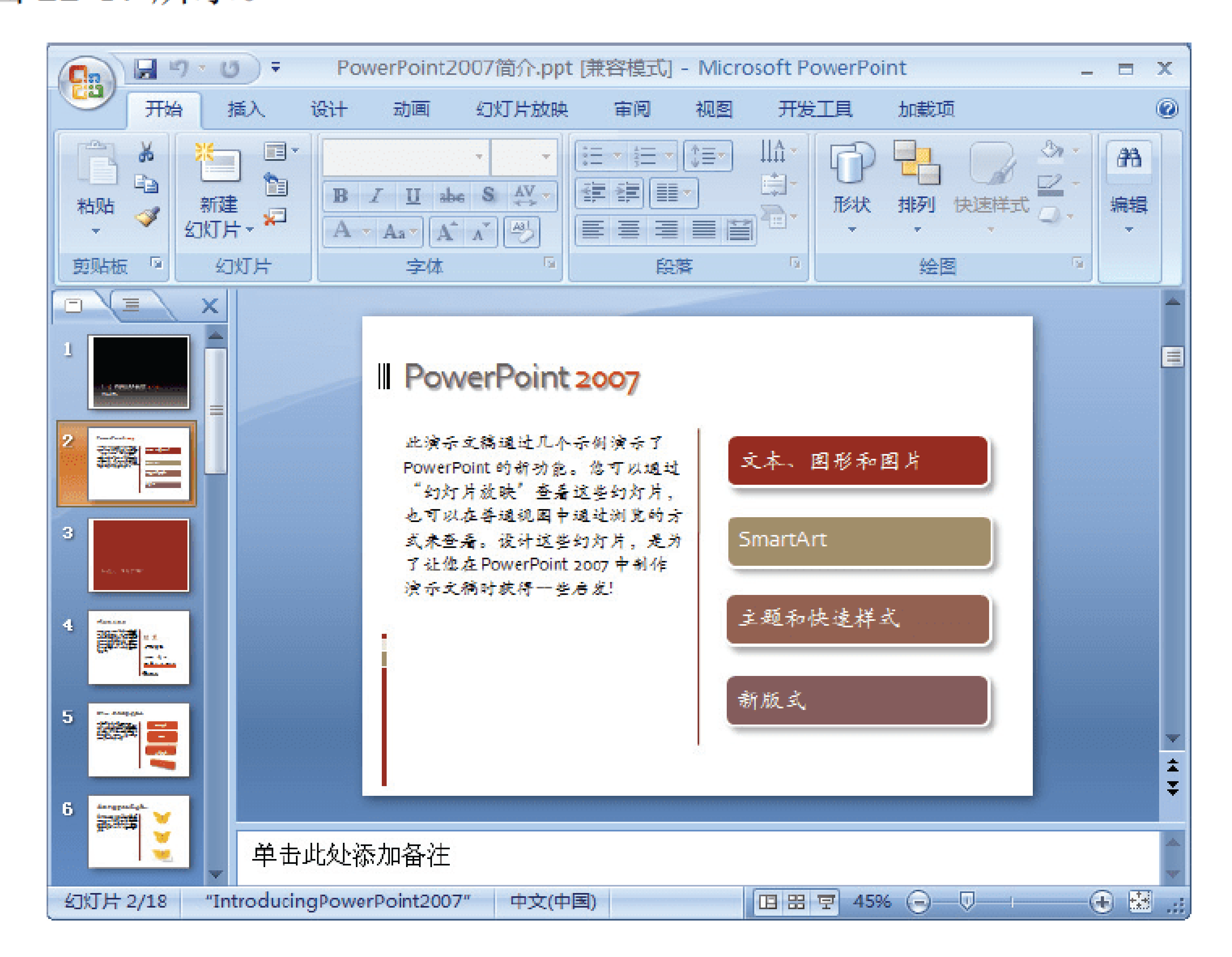


图 22-17 打开的演示文稿

22.3.3 创建演示文稿

在 Excel 中通过 VBA 创建 PowerPoint 演示文稿，可按以下顺序进行：

（1）使用 Presentations 对象的 Add 方法创建一个演示文稿。
（2）使用 Slides 对象的 Add 方法向演示文稿中插入一个幻灯片。
（3）向新插入的幻灯片中添加文字、图片等内容。
例如，以下代码创建一个演示文稿，并在其中添加一张幻灯片，然后保存该演示文稿。

```
With Presentations.Add
    .Slides.Add Index:=1, Layout:=ppLayoutTitle
    .SaveAs "Sample"
End With
```

以下代码将完整地演示通过工作表“合格证”中的数据创建演示文稿的方法。

```
Sub 创建演示文稿()
    Dim pptApp As PowerPoint.Application, pptPrs As PowerPoint.Presentation
    Dim ws1 As Worksheet , i As Long, j As Integer
    Dim str1 As String, strName As String , sFName As String, strFilt As String
    Dim strTitle As String, strInit As String

    strFilt = "PowerPoint 文件(*.ppt),*.ppt"
    strTitle = "保存 PowerPoint 幻灯片"
    strInit = "Excel 创建 PPT.ppt"

    Do                                              '要求用户必须输入文件名
        sFName = Application.GetSaveAsFilename(InitialFileName:=strInit, _
            filefilter:=strFilt, Title:=strTitle)
    Loop While sFName = "False" Or sFName = ""

    Set ws1 = ActiveWorkbook.Worksheets("合格证")
    Set pptApp = New PowerPoint.Application         '实例化 PPT 应用程序
    pptApp.Visible = True                           '显示 PPT 程序
    Set pptPrs = pptApp.Presentations.Add           '增加演示文稿
    With pptPrs
        For i = 1 To ws1.Range("A65536").End(xlUp).Row
            strName = ws1.Cells(i + 2, 2)           '获取车牌号
            str1 = ""
            For j = 3 To 13                         '生成字符串
                str1 = str1 & ws1.Cells(2, j) & ": " & ws1.Cells(i + 2, j)
                If j Mod 2 = 0 Then
                    str1 = str1 & vbCrLf & vbCrLf
                Else
                    str1 = str1 & "        "
                End If
            Next

            With .Slides.Add(Index:=i, Layout:=ppLayoutText).Shapes
                                                    '添加一个幻灯片
                .Title.TextFrame.TextRange = strName & " 出厂合格证"
                                                    '添加标题
                .Range(2).TextFrame.TextRange=str1 '添加正文
            End With
```

```
        Next
        .SaveAs sFName                                         '保存演示文稿
    End With
    Set ws1 = Nothing
    Set pptPrs = Nothing
    Set pptApp = Nothing
End Sub
```

以上代码首先让用户输入保存 PowerPoint 的文件名，接着从工作表“合格证”中获取相关数据，再使用 Slides 集合中的 Add 方法添加一张幻灯片，这样循环处理将每辆车的信息生成一张幻灯片，最后保存生成的幻灯片。如图 22-18 所示。

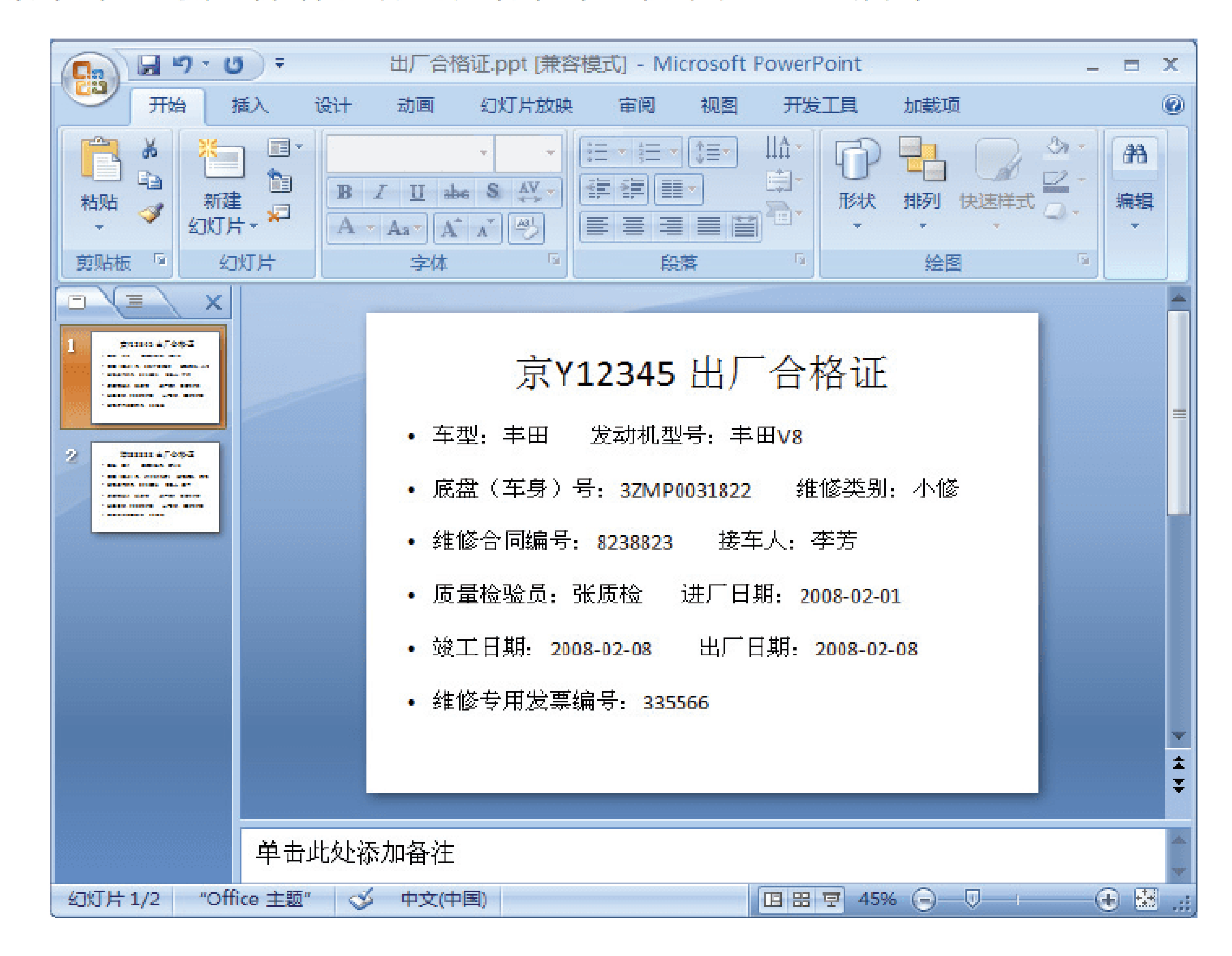


图 22-18　创建演示文稿

22.4　控制 Outlook 程序

使用 Outlook 对象模型提供的对象进行交互时，Excel 可以方便地控制这些 Outlook 应用程序。本节介绍用 Excel 控制 Outlook 发送邮件、读取 Outlook 中的邮件及附件等方法。

22.4.1　了解 Outlook 对象模型

Outlook 是 Office 办公套件的一个组件，是基于 Microsoft Exchange 的消息系统。若将 Outlook 安装到一台没有其他任何消息组件的计算机中，则安装程序会自动安装基于 MAPI 的 Microsoft Exchange 消息系统。

与任何一种消息系统一样，Microsoft Exchange 使用一个层次化的文件夹（Folder）集

合来存储数据。文件夹可以包含子文件夹（如收件箱、发件箱等）和条目（Item）（如邮件消息、约会、联系人和任务等）。

在 Outlook 对象模型中，Application 对象包含 NameSpace 对象；NameSpace 对象包含给定数据源（如 MAPI 消息库）中所有文件夹，即 Folder 对象集合；Folder 对象包含该文件夹中所有条目对象；每个条目对象包含用于对其进行控制的可编程对象。

在 Excel 中要使用 Outlook 对象模型中的对象，需在 VBE 工程中单击主菜单【工具】|【引用】命令，打开如图 22-19 所示的对话框，选中 Microsoft Outlook 12.0 Object libraary 复选框，将 Outlook 对象库添加到当前工程中。

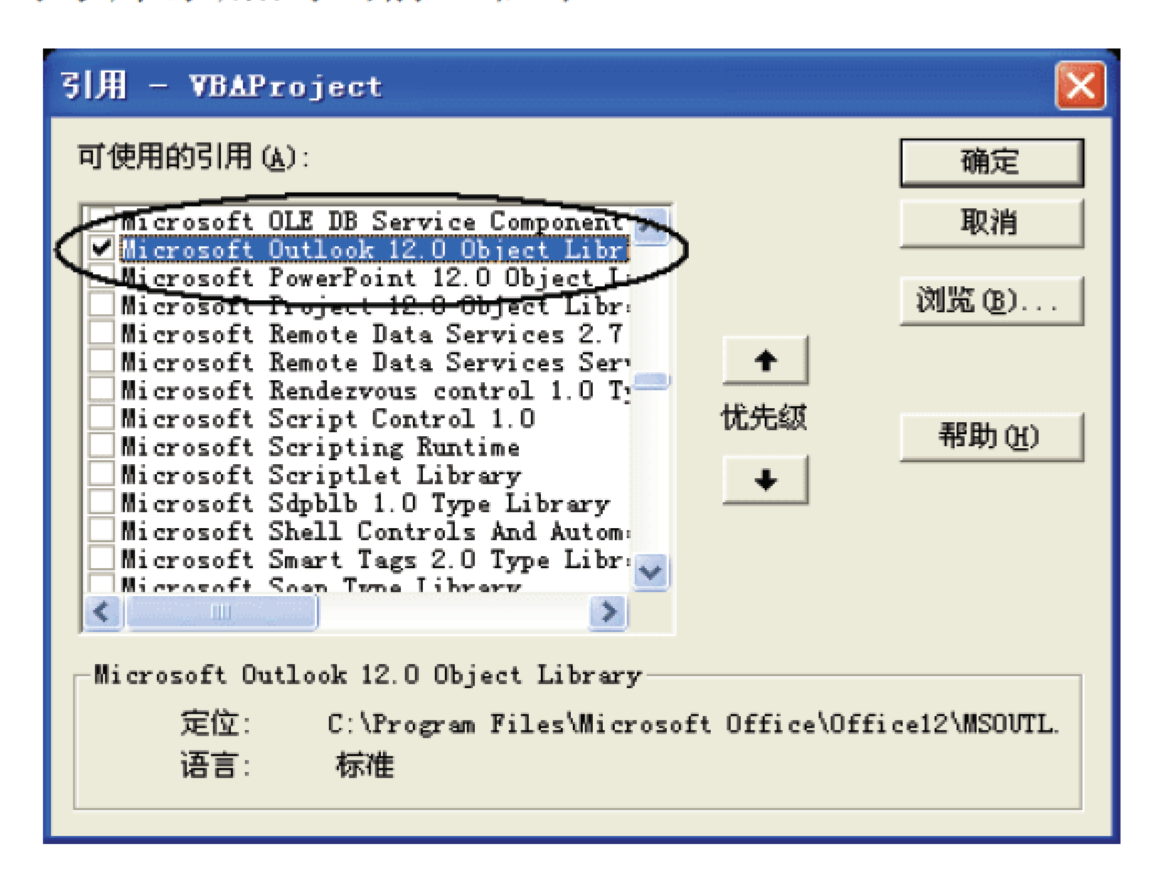


图 22-19　引用 Outlook 对象模型

下面简单介绍 Outlook 中的常用对象：

1. Application对象

Application 对象表示 Outlook 应用程序，它是 Outlook 对象模型中最高级的对象。此对象中最重要的成员包括下面几个。

- ❑ CreateItem 方法：该方法创建并返回新的 Outlook 项目。项目包括电子邮件、约会、联系人、任务、日记条目、便笺以及投递的项目和文档。
- ❑ Explorers 属性：返回 Explorers 集合对象，该对象包含表示所有已打开浏览器的 Explorer 对象。
- ❑ Inspectors 属性：该属性可用来访问显示单个项（如电子邮件或会议要求）内容的窗口。

2. Explorer对象

Explorer 对象表示显示包含项（如电子邮件、任务或约会）的文件夹内容的窗口。其包括可用来修改窗口的方法和属性，以及窗口更改时所引发的事件。

3. Inspector对象

Inspector 对象表示一个显示单个项（如电子邮件、任务或约会）的窗口。Inspector 对

象类包括可用来修改窗口的方法和属性，以及窗口更改时所引发的事件。

4．Folder对象

Folder 对象代表 Outlook 文件夹。

Folder 对象可包含其他 Folder 对象以及 Outlook 项目。使用 NameSpace 对象或另一个 Folder 对象的 Folders 属性可以返回 NameSpace 中或某个文件夹下面的文件夹集合。可以首先从顶级文件夹开始（如收件箱），然后结合使用 Folder.Folders 属性（该属性会返回层次结构中 Folder 对象下面的文件夹集合）和 Folders.Item 方法（该方法会返回 Folders 集合中的某个文件夹）导航嵌套文件夹。

使用 Folders.Add 方法可以将文件夹添加到 Folder 对象中。Add 方法有一个可选参数，用于指定在该文件夹中存储的项目类型。默认情况下，在不同文件夹中创建的文件夹会继承其父文件夹的类型。

5．MailItem对象

MailItem 对象表示收件箱文件夹中的邮件。使用 CreateItem 方法可创建表示新邮件的 MailItem 对象。使用 Items（index）可从收件箱文件夹返回单个 MailItem 对象，其中 index 是邮件的索引号或用于匹配邮件默认属性的值。

MailItem 对象的属性非常多，常用对象的属性含义如下面所述。

- UnRead：如果尚未打开（阅读）Outlook 项目，则该值为 True。
- SenderName：Outlook 项目的发件人的显示名称。
- SenderEmailAddress：Outlook 项目的发件人电子邮件地址。
- Subject：Outlook 项目的主题。

6．TaskItem

TaskItem 对象表示“任务”文件夹中的任务（要在指定时间内执行的分配的、代理的或自愿接受的任务）。

22.4.2 用 Outlook 发送邮件

通过 Outlook 对象提供的服务，在 Excel 中编写代码可发送邮件。例如，在如图 22-20 所示的工作表中输入邮件的相关内容，单击【发送】按钮，即可将邮件内容发送到输入的电子邮箱中去。

如果邮件中需要添加附件，单击【附件】按钮，在打开的对话框中选择添加的文件即可，如图 22-21 所示。

可按以下步骤完成邮件发送代码的编写：

（1）参照图 22-20 所示工作表设置邮件编写界面。

（2）按快捷键 Alt+F11 进入 VBE，并添加对 Outlook 对象模型的引用。

（3）编写发送邮件的 VBA 代码如下：

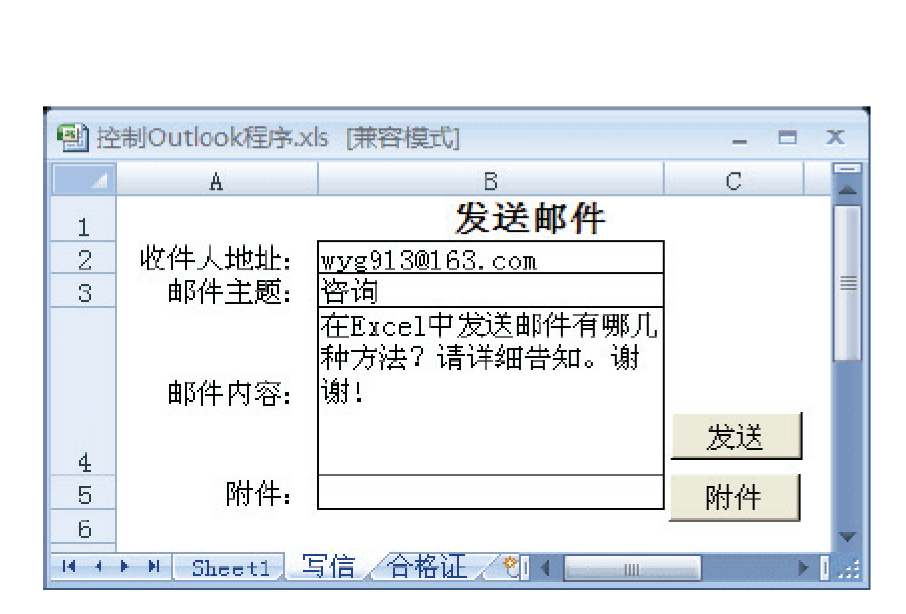


图 22-20　发送邮件

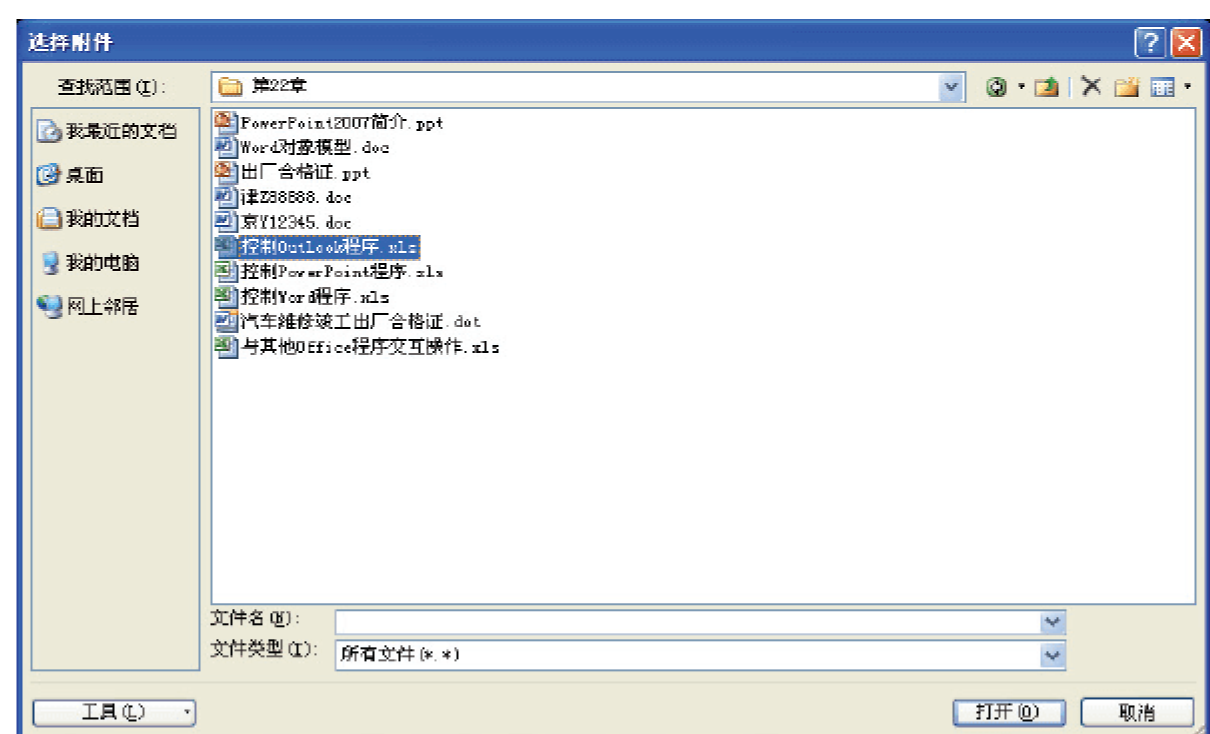


图 22-21　【选择附件】对话框

```
Sub 发送邮件()
    Dim strMail As String, strSubject As String
    Dim strBody As String, strAtt As String

    With ActiveSheet
        strMail = .Range("B2")                  '收件人地址
        strSubject = .Range("B3")               '邮件主题
        strBody = .Range("B4")                  '邮件内容
        strAtt = .Range("B5")                   '附件
    End With
    SendEmail strMail, strSubject, strBody, strAtt
    MsgBox "发送邮件完毕！", vbInformation + vbOKOnly, "提示"
End Sub
```

以上代码从工作表中获取收件人地址、邮件主题、邮件内容和附件，最后调用 SendEmail 子过程发送邮件。

（4）编写发送邮件的子过程 SendEmail 子过程，具体代码如下：

```
Function SendEmail(ByVal sMail As String, ByVal sSubject As String, _
        ByVal sBody As String, sAtt As String)
    Dim olApp As Object
    Dim olNameSpace As Object
    Dim olFolder As Object
    Dim olMail As Object
    Set olApp = CreateObject("Outlook.Application")
    Set olNameSpace = olApp.GetNamespace("MAPI")

    Set olFolder = olNameSpace.GetDefaultFolder(6)
    Set olMail = olApp.CreateItem(0)
    With olMail
        .subject = sSubject                         '邮件主题
        .Recipients.Add sMail                       '收件人地址
        .Body = sBody                               '邮件内容
        .Attachments.Add sAtt                       '邮件附件
        .Send
```

```
    End With
End Function
```

以上代码首先创建 Outlook 对象，再设置该对象各项参数，最后使用 Send 方法将邮件发送到目的地。

（5）在发送邮件时，还可为邮件添加附件，代码如下：

```
Sub 添加附件()
    Dim str1 As String
    str1 = Application.GetOpenFilename(Title:="选择附件")
    If str1 = "False" Or str1 = "" Then Exit Sub
    ActiveSheet.Range("B5") = str1
End Sub
```

添加附件的代码很简单，显示一个打开文件对话框，将用户选择的文件名显示在工作表对应的单元格即可。

22.4.3 获取 Outlook 保存的邮件

使用 Outlook 程序可从网络中接收邮件，将邮件接收到本地计算机后，不联网也可查看邮件内容。

要查看本地计算机中的邮件，一般也需要打开 Outlook 程序。在 Excel 中引用 Outlook 对象模型后，也可不打开 Outlook 程序就从 Outlook 中将邮件读取出来。

在 Outlook 中，使用 NameSpace 对象的 GetDefaultFolder 方法可返回 Folder 对象，表示当前配置文件所需类型的默认文件夹。该方法的语法格式如下：

```
表达式.GetDefaultFolder(FolderType)
```

参数 FolderType 用一个常量表示要返回的默认文件夹类型。常用的常量有以下几种。

- olFolderDrafts：草稿文件夹。
- olFolderInbox：收件箱文件夹。
- olFolderJournal：日记文件夹。
- olFolderOutbox：发件箱文件夹。
- olFolderSentMail：已发送邮件文件夹。

如果要获取 Outlook 中的邮件，需设置默认文件夹为 olFolderInbox（收件箱）。

下面的 VBA 代码将 Outlook“收件箱”中的内容导入 Excel 工作表，让用户不打开 Outlook 程序也能阅读邮件内容。

```
Sub 导入邮件()
    On Error Resume Next
    Dim olApp As New Outlook.Application, olNameSpace As Outlook.Namespace
    Dim olMailItem As Outlook.MailItem, olFolder As Outlook.Folder

    Set olNameSpace = olApp.GetNamespace("MAPI")
    Set olFolder = olNameSpace.GetDefaultFolder(olFolderInbox)
                                                         '收件箱
```

```
    For i = 1 To olFolder.Items.Count
        Set olMailItem = olFolder.Items(i)   '获取一个邮件项目
        With olMailItem
            ActiveSheet.Cells(i + 2, 1) = IIf(.UnRead, "未读", "已读")
                                             '状态
            ActiveSheet.Cells(i + 2, 2) = .SenderName & "(" & .SenderEmail
            Address & ")"
                                             '发件人
            ActiveSheet.Cells(i + 2, 3) = .Subject
                                             '主题
        End With
    Next
End Sub
```

第 23 章　处 理 文 件

文件是计算机程序设计中的一个重要概念。所谓文件，是指保存在计算机磁盘中的数据的集合。操作系统对数据的管理是以文件为单位进行的。VBA 为处理文件提供了很多的命令，本章将详细介绍 VBA 处理文件的方法。

23.1　常用文件操作语句

VBA 提供了很多语句进行文件操作，这是传统的操作文件的语句。这些文件操作的语句可分为 3 类，分别为文件管理、文件创建、文件读写。

23.1.1　文件管理语句

文件管理语句主要用来对已有文件进行查找、复制、重命名和删除等操作。

1．查找文件——Dir函数

Dir 函数返回一个文件名或文件夹名称，它必须与指定的模式或文件属性、磁盘卷标相匹配。其语法格式如下：

```
Dir[(pathname[, attributes])]
```

该函数的两个参数都可省略，各参数的含义如下所述。

- pathname：用来指定查找的内容，可能包含文件夹或驱动器。如果没有找到 pathname，则会返回零长度字符串（""）。
- attributes：用来指定文件属性，将返回指定属性的文件或文件夹。如果省略此参数，则会返回匹配 pathname 但不包含属性的文件。

attributes 参数可设为下面的几种常量。

- vbNormal：指定没有属性的文件；
- vbReadOnly：指定无属性的只读文件；
- vbHidden：指定无属性的隐藏文件；
- vbSystem：指定无属性的系统文件；
- vbVolume：指定卷标文件；
- vbDirectory：指定无属性文件及其路径和文件夹。

Dir 会返回匹配 pathname 的第一个文件名。若想得到其他匹配 pathname 的文件名，

可再一次调用 Dir，且不需使用参数。如果已没有合乎条件的文件，则 Dir 会返回一个零长度字符串（""）。一旦返回值为零长度字符串，并需再次调用 Dir 时，就必须指定 pathname，否则会产生错误。

提示：由于文件名并不会以特别的次序来返回，所以可以将文件名存储在一个数组中，然后再对这个数组排序。

例如，以下代码将在 C:\Windows 文件夹中查找扩展名为.ini 的文件，如果该文件夹中有超过一个*.ini 文件存在，函数将返回第一个按条件找到的文件名。

```
MyFile = Dir("C:\WINDOWS\*.ini")
```

2．重命名——Name语句

使用 Name 语句重新命名一个文件或文件夹。该语句的语法格式如下：

```
Name oldpathname As newpathname
```

Name 语句的语法包含下面两部分的内容。

- oldpathname：指定已存在的文件名和位置，可以包含文件夹或驱动器。
- newpathname：指定新的文件名和位置，可以包含文件夹或驱动器。而由 newpathname 所指定的文件名不能存在。

Name 语句重新命名文件，并将其移动到一个不同的文件夹中。Name 可在不同驱动器之间移动文件。但当 newpathname 和 oldpathname 都在相同的驱动器中时，只能重新命名已经存在的文件夹。

注意：在一个已打开的文件上使用 Name，将会产生错误。所以必须在改变名称之前，先关闭打开的文件。Name 参数不能包括多字符（*）和单字符（?）的通配符。

3．删除文件——Kill语句

使用 Kill 语句可从磁盘中删除文件，其语法格式如下：

```
Kill pathname
```

参数 pathname 是用来指定一个文件名的字符串表达式。pathname 可以包含文件夹或驱动器。

Kill 支持多字符（*）和单字符（?）的通配符来指定多重文件。例如，下列语句删除当前文件夹内的所有 TEXT 文件：

```
Kill  "*.TXT"
```

注意：如果使用 Kill 来删除一个已打开的文件，则会产生错误。

4．删除文件夹——RmDir语句

使用 RmDir 语句可删除一个文件夹，其语法格式如下：

```
RmDir path
```

参数 path 是一个字符串表达式，用来指定要删除的目录或文件夹。path 可以包含驱动器。如果没有指定驱动器，则 RmDir 会在当前驱动器上删除目录或文件夹。

注意：如果想要使用 RmDir 来删除一个含有文件的文件夹，则会发生错误。在试图删除文件夹之前，可以先使用 Kill 语句来删除所有文件。

5．复制文件

FileCopy 语句用来进行文件的复制操作，该语句的语法格式如下：

```
FileCopy source, destination
```

FileCopy 语句的语法含有以下这些命名参数。

- source：用来表示要被复制的文件名。其可以包含文件夹或驱动器。
- destination：用来指定要复制的目标文件名。其可以包含文件夹或驱动器。

注意：如果想要对一个已打开的文件使用 FileCopy 语句，则会产生错误。

6．操作文件夹的语句

在 Windows 操作系统中，有一个当前文件夹的概念。在 VBA 中对文件进行操作时，如果未指定路径，则都是指操作当前文件夹中的文件。下面几种是常用的操作文件夹的语句。

- ChDir：改变当前文件夹。
- ChDrive：改变磁盘。
- MkDir：新建文件夹。
- RmDir：删除文件夹。
- CurDir：返回当前路径。

23.1.2　创建文件语句

使用 VBA 语句可直接创建文本文件，下面简单介绍创建文件需要用到的语句。

1．可用文件号——FreeFile函数

使用 VBA 传统语句操作文件时，首先必须打开文件。在使用 Open 语句打开文件时，必须为文件指定一个文件号。利用该文件号才可以从文件中读取数据或向文件中写入数据。

为了得到一个有效的文件号，可使用 FreeFile 函数，该函数返回一个整型值，代表下一个可供 Open 语句使用的文件号。

该函数的语法格式如下：

```
FreeFile[(rangenumber)]
```

参数 rangenumber 可省略，它指定一个范围，以便返回该范围之内的下一个可用文件号。指定 0（默认值）则返回一个介于 1～255 之间的文件号。指定 1 则返回一个介于 256～

511 之间的文件号。

2. 打开文件——Open语句

对文件做任何 I/O（输入/输出）操作之前都必须先打开文件。Open 语句分配一个缓冲区供文件进行 I/O 之用，并决定缓冲区所使用的访问方式。

使用 Open 语句打开文件，其语法格式如下：

```
Open pathname For mode [Access access] [lock] As [#]filenumber [Len=
reclength]
```

Open 语句的语法包含下面几部分内容。

- ❑ pathname：指定文件名，该文件名可能还包括目录、文件夹及驱动器。
- ❑ mode：指定文件方式，可设置为 Append、Binary、Input、Output、或 Random 方式。如果未指定方式，则以 Random 访问方式打开文件。
- ❑ Access：说明打开的文件可以进行的操作，有 Read、Write、或 Read Write 操作。
- ❑ lock：说明限定于其他进程打开的文件的操作，有 Shared、Lock Read、Lock Write、和 Lock Read Write 操作。
- ❑ filenumber：一个有效的文件号，范围在 1～511 之间。使用 FreeFile 函数可得到下一个可用的文件号。
- ❑ reclength：小于或等于 32 767（字节）的一个数。对于用随机访问方式打开的文件，该值就是记录长度。对于顺序文件，该值就是缓冲字符数。

如果 pathname 指定的文件不存在，在用 Append、Binary、Output、或 Random 方式打开文件时，可以建立该文件。

技巧： 在 Binary、Input 和 Random 方式下可以用不同的文件号打开同一个文件，而不必先将该文件关闭。在 Append 和 Output 方式下，如果要用不同的文件号打开同一个文件，则必须在打开文件之前先关闭该文件。

3. 关闭文件——Close语句

在文件使用结束后，必须使用 Close 关闭文件，释放文件号。其语法格式如下：

```
Close [filenumberlist]
```

参数 filenumberlist 为一个或多个文件号，其中 filenumber 为任何有效的文件号。若省略 filenumberlist，则将关闭 Open 语句打开的所有活动文件。

当关闭 Output 或 Append 打开的文件时，将属于此文件的最终输出缓冲区写入操作系统缓冲区。所有与该文件相关联的缓冲区空间都被释放。

在执行 Close 语句后，文件与其文件号之间的关联将终结。

23.1.3 向文件中写入数据

使用 Open 语句打开或创建一个文件后，就可以使用返回的文件号向文件中写入数据。

1. Print #语句

使用 Print #语句可将格式化显示的数据写入顺序文件中。该语句的语法格式如下：

```
Print #filenumber, [outputlist]
```

Print #语句的语法包含下面几部分的内容。

- ❑ filenumber：为使用 Open 语句打开文件时获取的文件号。
- ❑ outputlist：为要输入到文件的表达式或表达式列表。

输出表达式 outputlist 可设置为以下内容：

```
[{Spc(n) | Tab[(n)]}] [expression] [charpos]
```

- ❑ Spc(n)：用来在输出数据中插入空白字符，而 n 指的是要插入的空白字符数。
- ❑ Tab(n)：用来将插入点定位在某一绝对列号上，这里“n”是列号。使用无参数的 Tab 将插入点定位在下一个打印区的起始或位置。
- ❑ expression：要打印的数值表达式或字符串表达式。
- ❑ charpos：指定下一个字符的插入点。使用分号将插入点定位在上一个显示字符之后。用 Tab(n)将插入点定位在某一绝对的列号上，用无参数的 Tab 将插入点定位在下一个打印区的起始处。如果省略 charpos，则在下一行打印下一个字符。

2. Write #语句

使用 Write #语句可将变量中的数据写入文件中。其语法格式如下：

```
Write #filenumber, [outputlist]
```

Write # 语句的语法包含下面几部分的内容。

- ❑ filenumber：表示需要写入数据的文件的文件号。
- ❑ outputlist：要写入文件的数值表达式或字符串表达式，用一个或多个逗号将这些表达式分界。

如果省略 outputlist，并在 filenumber 之后加上一个逗号，则会将一个空白行打印到文件中。多个表达式之间可用空白、分号或逗号隔开。空白和分号等效。

23.1.4 从文件中读出数据

使用 Open 语句打开文件后，可通过 VBA 相关语句从文件中读出数据，并赋值给变量进行处理。

1. Input #语句

使用 Input #语句从已打开的顺序文件中读出数据并将数据指定给变量。该语句的语法格式如下：

```
Input #filenumber, varlist
```

该语句的语法包括以下几部分的内容。

- ❑ filenumber：使用 Open 语句打开的文件号。
- ❑ varlist：用逗号分界的变量列表，将文件中读出的值分配给这些变量；这些变量不可能是一个数组或对象变量。

文件中数据项目的顺序必须与 varlist 中变量的顺序相同，而且与相同数据类型的变量匹配。如果变量为数值类型，而数据不是数值类型，则指定变量的值为零。

在输入数据项目时，如果已到达文件结尾，则会终止输入，并产生一个错误。

2．EOF函数

使用 Open 语句打开文件后，文件的当前位置位于文件首部，使用 Input #语句读取文件的部分数据后，文件的当前位置自动向后移动，下次使用 Input #语句时可读取后续的数据。

为了避免因试图在文件结尾处进行输入而产生的错误，可使用 EOF 函数判断当前位置是否已到了文件的结尾。EOF 函数的语法格式如下：

```
EOF(filenumber)
```

其中，参数 filenumber 为一个文件号。

3．Line Input #语句

使用 Line Input #语句从已打开的顺序文件中读出一行并将它分配给 String 变量。该语句的语法格式如下：

```
Line Input #filenumber, varname
```

Line Input #语句的语法包括以下几部分的内容。

- ❑ filenumber：使用 Open 语句打开的文件号。
- ❑ varname：保存文件值的变量名。

Line Input #语句一次只从文件中读出一个字符，直到遇到回车符（Chr(13)）或回车换行符（Chr(13) + Chr(10)）为止。回车换行符将被跳过，而不会被附加到字符串上。

23.2　文件对象模型

文件对象模型 FSO 的全称是 File System Object，该模型提供了一个基于对象的工具来处理文件夹和文件。通过采用面向对象编程的语法（使用 object.method 这种方式），将一系列文件和文件夹的操作通过调用对象本身的属性和方法来实现。

23.2.1　文件对象模型简介

FSO 对象模型不仅可以像使用传统文件操作语句那样实现文件的创建、改变、移动和删除，而且可以检测是否存在指定的文件夹，以及文件夹位于磁盘上的什么位置。FSO 对象模型还可以获取关于文件和文件夹的信息，例如，名称、创建日期或最近修改日期等以

及当前系统中使用的驱动器的信息。

FSO 对象模型包含在 Scripting 类型库(Scrrun.dll)中，它同时包含了 FileSystemObject、Drive、Folder、File 和 TextStream 等 5 个对象。

- FileSystemObject 对象是 FSO 对象模型中最主要的对象，它提供了一套完整的可用于创建、删除文件和文件夹；收集驱动器、文件夹、文件相关信息的方法。
- Drive 对象用来收集驱动器的信息，如可用磁盘空间或驱动器的类型等。
- Folder 对象用于创建、删除或移动文件夹，同时可以进行向系统查询文件夹的路径等操作。
- File 对象的基本操作和 Folder 大致相同，所不同的是 Files 的操作主要是针对磁盘上的文件进行的。
- TextStream 对象则是用来完成对文件的读写等操作。

注意：FSO 对象模型提供的方法是冗余的，FileSystemObject 对象的方法是直接作用于其余对象，所以 FileSystemObject 中的很多方法与其他 4 种对象的方法有重复，读者可根据实际情况或上下文环境来选择使用某个对象提供的方法。

23.2.2　引用 FSO 对象

在 VBA 中要使用 FSO 模型中的对象，需要先将 Scripting 类型库加入工程中，具体方法为：

（1）在 VBE 中单击主菜单【工具】|【引用】命令，打开【引用】对话框。

（2）在【引用】对话框中找到 Microsoft Scripting Runtime 选项，单击选中，之后单击【确定】按钮，如图 23-1 所示。



图 23-1　引用 FSO 对象模型

通过以上步骤即可将 Scripting 类型库加入到当前工程中。即可在 VBA 程序中创建 FileSystemObject 对象，并引用该对象库中的对象对文件进行操作。

例如，可使用以下语句声明 FSO 对象变量。

```
Dim fso As New Scripting.FileSystemObject
```

以上声明语句使用了 New 关键字，在第一次引用该变量时将新建该对象的实例，因此

不必使用 Set 语句来给该对象引用赋值。若不使用 New 关键字，则需使用以下两条语句完成上面的功能：

```
Dim fso As Scripting.FileSystemObject
Set fso = New Scripting.FileSystemObject
```

提示：将 FSO 对象库添加到工程中以后，即可使用相关对象提供的属性和方法操作驱动器、文件和文件夹。具体内容将在本章后续的实例中进行介绍。

23.3　获得文件信息

在处理文件时，可通过传统的 VBA 语句或 FSO 对象模型获取磁盘、文件的相关信息。

23.3.1　获取磁盘信息

在 FSO 对象模型中，通过 Drive 对象提供对磁盘驱动器或网络共享的属性的访问。

Drive 对象没有方法，其应用都是通过属性表现出来的，该对象的各属性分别如下面所述。

- AvailableSpace：返回在指定的驱动器或网络共享上的用户可用的空间容量。
- DriveLetter：返回某个指定本地驱动器或网络驱动器的字母，该属性为只读。
- DriveType：返回指定驱动器的磁盘类型。
- FileSystem：返回指定驱动器使用的文件系统类型。
- FreeSpace：返回指定驱动器上或共享驱动器可用的磁盘空间，该属性为只读。
- IsReady：确定指定的驱动器是否准备好。
- Path：返回指定文件、文件夹、或驱动器的路径。
- RootFolder：返回一个 Folder 对象，该对象表示一个指定驱动器的根文件夹。其为只读属性。
- SerialNumber：返回用于唯一标识磁盘卷标的十进制序列号。
- ShareName：返回指定驱动器的网络共享名。
- TotalSize：以字节为单位，返回驱动器或网络共享的总空间大小。
- VolumeName：设置或返回指定驱动器的卷标名。

使用以下代码可获取磁盘信息，并将显示到工作表中：

```
Sub 获取磁盘信息()
    Dim fso As New Scripting.FileSystemObject, oDrive As Scripting.Drive
    Dim i As Integer, sType As String, sh As Worksheet

    Set sh = Worksheets("磁盘信息")
    sh.Cells.Clear
    i = -1
    For Each oDrive In fso.Drives
```

```
        i = i + 2
        sh.Cells(i, 1) = oDrive.DriveLetter & "盘信息: "
        sh.Cells(i, 1).Font.Bold = True
        i = i + 1
        With oDrive
            Select Case .DriveType
                Case 0: sType = "不明"
                Case 1: sType = "可移动"
                Case 2: sType = "硬盘"
                Case 3: sType = "网络驱动器"
                Case 4: sType = "CD-ROM"
                Case 5: sType = "RAM 驱动器"
            End Select
            sh.Cells(i, 1) = "驱动器类型: " & sType
            i = i + 1
            sh.Cells(i, 1) = "驱动器是否可用: " & IIf(.IsReady, "可用", "不可用")
            i = i + 1
            If .IsReady Then
               sh.Cells(i, 1) = "驱动器容量: " & Round(.TotalSize / 1024 / 1024,
               2) & "MB"
                i = i + 1
       sh.Cells(i, 1) = "驱动器可用空间: " & Round(.AvailableSpace / 1024 / 1024,
       2) & "MB"
                i = i + 1
         sh.Cells(i, 1) = "驱动器剩余空间: " & Round(.FreeSpace / 1024 / 1024,
         2) & "MB"
                i = i + 1
                sh.Cells(i, 1) = "驱动器文件系统: " & .FileSystem
                i = i + 1
               sh.Cells(i, 1) = "驱动器的 Volume 名称: " & .VolumeName
           End If
        End With
    Next
    sh.Columns(1).AutoFit
    Set oDrive = Nothing
    Set fso = Nothing
End Sub
```

以上代码从 Drives 集合分别获取计算中的每一个磁盘，再通过 Drive 对象获取每个磁盘的属性，并显示在工作表中。运行的结果如图 23-2 所示。

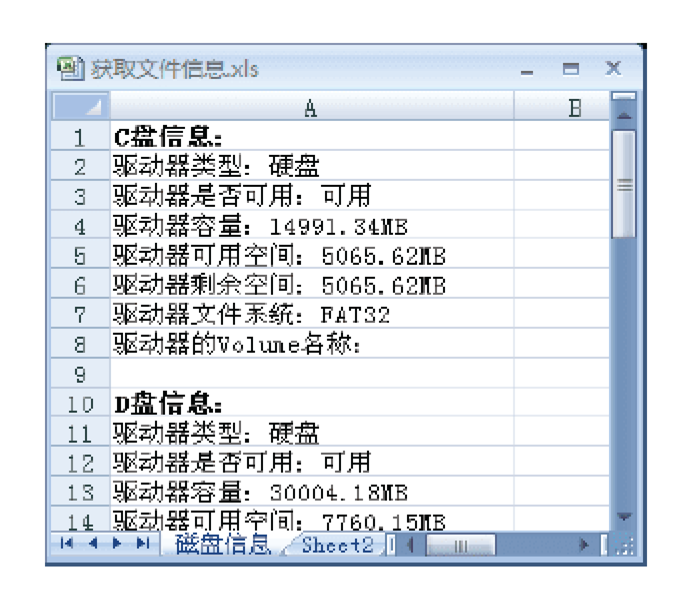


图 23-2　获取磁盘信息

23.3.2　查看文件信息

要查看文件信息，可以使用传统 VBA 语句和 FSO 对象模型两种方式。

1．使用传统VBA语句查看文件信息

使用 VBA 提供的下面几种函数可获取文件的属性。

- FileLen 函数

该函数返回一个代表文件长度的长整型值，单位为字节，其语法格式如下：

```
FileLen(pathname)
```

其中，参数 pathname 是用来指定一个文件名的字符串表达式。pathname 可以包含文件夹或驱动器。

当调用 FileLen 函数时，如果所指定的文件已经打开，则返回的值是该文件在打开前的大小。若要取得一个打开文件的长度大小，可使用 LOF 函数。

- GetAttr 函数

该函数返回一个文件或文件夹的属性，其语法格式如下：

```
GetAttr(pathname)
```

pathname 参数是用来指定一个文件名的字符串表达式。pathname 可以包含文件夹或驱动器。

该函数的返回值可能是下面各数值中任意数值之和。

- vbNormal：值为 0，表示常规文件；
- vbReadOnly：值为 1，表示只读文件；
- vbHidden：值为 2，表示隐藏文件；
- vbSystem：值为 4，表示系统文件；
- vbDirectory：值为 16，表示为文件夹；
- vbArchive：值为 32，表示上次备份以后，文件已经改变；

若要判断是否设置了某个属性，可在 GetAttr 函数与想要得知的属性值之间使用 And 运算符与逐位比较。如果所得的结果不为零，则表示设置了这个属性值。例如，在下面的 And 表达式中，如果档案（Archive）属性没有设置，则返回值为 0：

```
Result = GetAttr(FName) And vbArchive
```

如果文件的档案属性已设置，则返回非零的数值。

- FileDateTime 函数

该函数返回一个日期值（Date），此为一个文件被创建或最后修改后的日期和时间，其语法格式如下：

```
FileDateTime(pathname)
```

其中，参数 pathname 是用来指定一个文件名的字符串表达式，其可以包含文件夹或驱动器。

使用以下代码可显示指定文件的属性。

```
Sub 查看文件属性()
    Dim sFileName As String, sType As String
    Dim i As Integer, iType As Integer
    sFileName = Application.GetOpenFilename(, , "选择文件")
    If sFileName = "False" Then Exit Sub        '用户选择“取消”则退出程序
    With Worksheets("文件属性")
        .Cells.Clear                            '清除工作表内容
        .Cells(1, 1) = "文件名: "
        .Cells(1, 2) = Mid(sFileName, InStrRev(sFileName, "\") + 1)
                                                '取文件名
        .Cells(2, 1) = "文件大小: "
        .Cells(2, 2) = FileLen(sFileName) & "Byte"
        .Cells(3, 1) = "文件类型: "
        iType = GetAttr(sFileName)
        If iType And vbNormal = 0 Then
            sType = "常规文件"
        ElseIf iType And vbReadOnly Then
            sType = "只读文件"
        ElseIf iType And vbHidden Then
            sType = "隐藏文件"
        ElseIf iType And vbSystem Then
            sType = "系统文件"
        ElseIf iType And vbArchive Then
            sType = "备份文件"
        ElseIf iType And vbDirectory Then
            sType = "文件夹"
        End If
        .Cells(3, 2) = sType
        .Cells(4, 1) = "最后修改时间: "
        .Cells(4, 2) = FileDateTime(sFileName)
        .Columns("A:B").AutoFit
    End With
End Sub
```

执行以上代码，首先将显示如图 23-3 所示对话框，在该对话框中选择一个文件后，单击【打开】命令，即可在工作表“文件属性”中显示选择文件的相关属性，如图 23-4 所示。

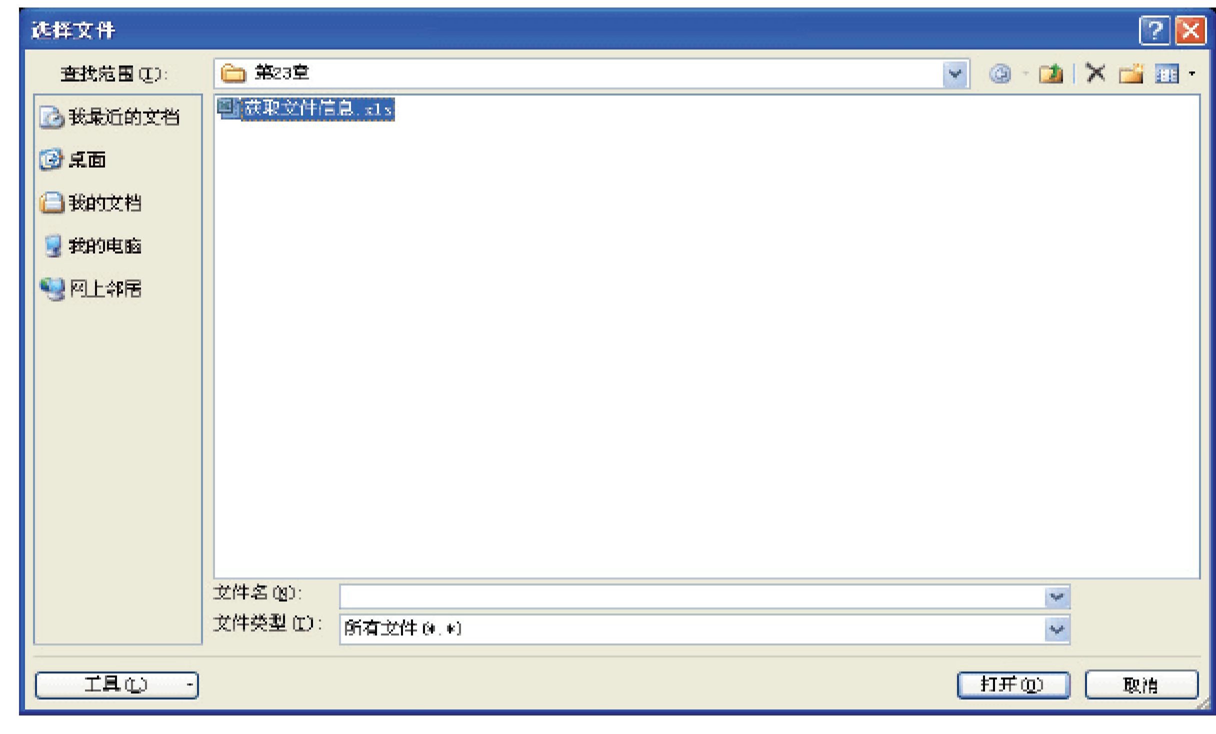


图 23-3 【选择文件】对话框

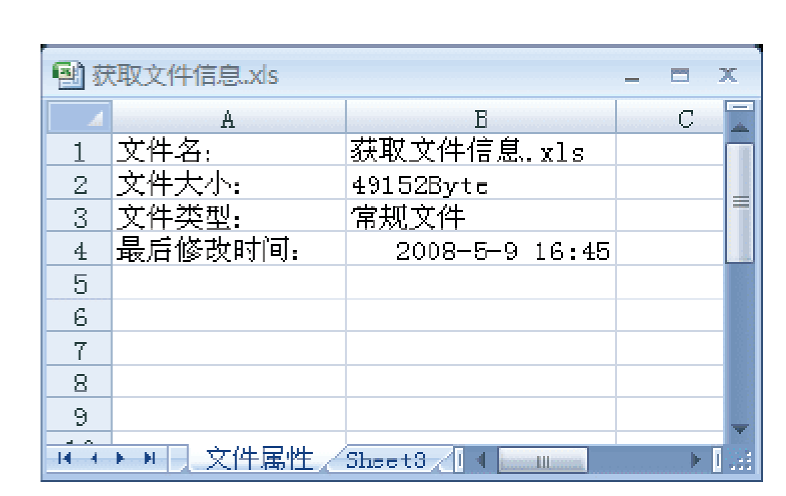


图 23-4 文件属性

2．使用FSO对象模型查看文件属性

在 FSO 对象模型中，File 对象提供了对文件属性的访问。可以通过 GetFile 方法从 FileSystemObject 对象中得到一个 File 对象引用。File 对象的属性如下所述。

- Attributes：返回文件的属性。可以是下列值中的一个或其组合：Normal（0）、ReadOnly（1）、Hidden（2）、System（4）、Volume（9）、Directory（16）、Archive（32）。
- DateCreated：返回该文件夹的创建日期和时间。
- DateLastAccessed：返回最后一次访问该文件的日期和时间。
- DateLastModified：返回最后一次修改该文件的日期和时间。
- Drive：返回该文件所在的驱动器的 Drive 对象。
- Name：设定或返回文件的名字。
- ParentFolder：返回该文件的父文件夹的 Folder 对象。
- Path：返回文件的绝对路径，可使用长文件名。
- ShortName：返回 DOS 风格的 8.3 形式的文件名。
- ShortPath：返回 DOS 风格的 8.3 形式的文件绝对路径。
- Size：返回该文件的大小（字节）。
- Type：如果可能，返回一个文件类型的说明字符串（例如：Text Document 表示.txt 文件）。

使用 FSO 对象模型查看文件属性的 VBA 代码如下：

```
Sub 用 FSO 显示文件属性()
    Dim fso As New FileSystemObject, oFile As File
    Dim sType As String, str1 As String, sFileName As String
    sFileName = Application.GetOpenFilename(, , "选择文件")
    If sFileName = "False" Then Exit Sub    '用户选择“取消”则退出程序
    Set oFile = fso.GetFile(sFileName)
    With oFile
        If (.Attributes And Normal) = Normal Then sType = sType & "普通"
        If (.Attributes And ReadOnly) = ReadOnly Then sType = sType & "只读"
        If (.Attributes And Hidden) = Hidden Then sType = sType & "隐藏"
        If (.Attributes And System) = System Then sType = sType & "系统"
        If (.Attributes And Directory) = Directory Then sType = sType &
        "文件夹"
        If (.Attributes And Archive) = Archive Then sType = sType & "存档"
    End With
    With Worksheets("文件属性")
        .Range("A1:B8").ClearContents
        .Cells(1, 1) = "文件名称："
        .Cells(1, 2) = oFile.Name
        .Cells(2, 1) = "短文件名："
        .Cells(2, 2) = oFile.ShortName
        .Cells(3, 1) = "文件路径："
        .Cells(3, 2) = oFile.Path
        .Cells(4, 1) = "文件类型："
```

```
        .Cells(4, 2) = sType
        .Cells(5, 1) = "文件大小: "
        .Cells(5, 2) = Round(oFile.Size / 1024, 2) & "KB"
        .Cells(6, 1) = "所在驱动器: "
        .Cells(6, 2) = oFile.Drive
        .Cells(7, 1) = "所在文件夹: "
        .Cells(7, 2) = oFile.ParentFolder
        .Cells(8, 1) = "创建时间: "
        .Cells(8, 2) = oFile.DateCreated
        .Columns("A:B").AutoFit
    End With
    Set oFile = Nothing
    Set fso = Nothing
End Sub
```

23.4 文件管理

在 Excel 中，编写 VBA 代码可方便地复制、删除文件，为文件改名，分离出文件的名称和扩展名等操作。

23.4.1 文件是否存在

在对文件操作之前，最好使用代码判断指定的文件是否存在。使用 Dir 函数可完成这种判断。例如，以下函数可判断文件是否存在：

```
Function FileExists(fname) As Boolean
    FileExists = Dir(fname) <> ""
End Function
```

函数的参数为要判断的文件名，当 fname 参数指定的文件存在，函数返回 True，否则返回 False。

以下代码调用 FileExists 函数判断输入的文件名是否存在：

```
Sub 判断文件()
    Dim sFileName As String
    sFileName = Application.InputBox(prompt:="请输入文件名称: ", _
        Default:=ThisWorkbook.FullName, Title:="输入文件名称", Type:=2)
    If sFileName = "False" Then Exit Sub    '用户选择"取消"则退出程序
    If sFileName = "False" Then Exit Sub    '用户选择"取消"则退出程序
    If FileExists(sFileName) Then
        MsgBox "文件 " & sFileName & " 存在! "
    Else
        MsgBox "文件 " & sFileName & " 不存在! "
    End If
End Sub
```

如果不使用自定义的函数，也可使用 FSO 对象的 FileExists 方法判断文件是否存在。如果使用 FSO 对象模型的方法来改写以上代码，具体代码为：

```
Sub FSO 判断文件()
    Dim fso As New FileSystemObject, sFileName As String
    sFileName = Application.InputBox(prompt:="请输入文件名称：", _
        Default:=ThisWorkbook.FullName, Title:="输入文件名称", Type:=2)
    If sFileName = "False" Then Exit Sub    '用户选择"取消"则退出程序
    If fso.FileExists(sFileName) Then
        MsgBox "文件 " & sFileName & " 存在！"
    Else
        MsgBox "文件 " & sFileName & " 不存在！"
    End If
    Set fso = Nothing
End Sub
```

23.4.2　复制文件

使用 FileCopy 语句可完成文件的复制操作。同时，FSO 对象也提供了一个名为 FileCopy 的方法，可用来进行文件的复制操作。

1．用VB语句复制文件

使用 FileCopy 语句复制文件时，需要确定源文件和目标文件。如果使用 InputBox 对话框让用户输入，则用户需要记住源文件的位置，如果输入错误的路径，将使复制操作失败。下面的代码使用 Application 对象的 GetOpenFilename 方法显示【打开】对话框，以方便用户在不同的文件夹中选择源文件。同样，使用 GetSaveAsFilename 方法，可让用户输入保存文件的名称。具体代码如下：

```
Sub 复制文件()
    Dim SourceFile As String, DestinationFile As String
    On Error GoTo Err1
    SourceFile = Application.GetOpenFilename(, , "选择源文件")
    If SourceFile = "False" Then Exit Sub   '用户选择"取消"则退出程序
    DestinationFile = Application.GetSaveAsFilename(, , , "输入目标文件名")
    If DestinationFile = "False" Then Exit Sub
    FileCopy SourceFile, DestinationFile
    MsgBox "复制成功！"
    Exit Sub
Err1:
    If Err.Number <> 0 Then
        MsgBox "无法复制该文件！"
    End If
End Sub
```

以上代码中，使用了错误捕捉语句，在复制文件过程中出现不可预测的错误时，将显示出错提示信息。

2．用FSO对象的方法复制文件

使用 FSO 对象模型中的 CopyFile 方法也能完成文件的复制操作。该方法可将一个或多个文件从某位置复制到另一位置，其语法格式如下：

```
object.CopyFile source, destination[,overwritefiles]
```

各参数的含义如下面所述。

- source：指定源文件，如果要复制的是一个或多个文件时，文件名中可以有通配符。
- destination：目标位置，表示从 source 复制文件到该位置。
- overwritefiles：设置是否覆盖现有文件。如果是 True，则覆盖文件；如果是 False，则不覆盖现有文件。默认值是 True。要注意，无论 overwrite 设置为何值，只要设置 destination 为只读属性，CopyFile 操作就无法完成。

在 source 参数的路径的最后一个组成部分中，可使用通配符。例如，可以使用：

```
fso.CopyFile "D:\DOC\*.doc", "E:\BAK\"
```

如果 source 包含通配符或 destination 以路径分隔符（\）结束，则假定 destination 是现有文件夹，复制匹配文件到该文件夹。否则，假定 destination 为要创建的文件。在任一种情况下，复制单个文件时，会出现以下 3 种情况。

- 如果 destination 不存在，则复制 source。
- 如果 destination 是已经存在的文件，当 overwrite 为 False 时会出现错误。否则，复制 source 覆盖现有文件。
- 如果 destination 是文件夹，则将源文件复制到该文件夹中。

提示： 如果 source 使用通配符，但并没有相匹配的文件时，则会出现错误。CopyFile 方法将在遇到出现的第一个错误时停止。该方法不会撤销错误发生前所作的任何更改。

以下代码使用 FSO 对象的 FileCopy 方法完成文件的复制操作。

```
Sub FSO复制文件()
    Dim fso As New Scripting.FileSystemObject
    Dim SourceFile As String, DestinationFile As String
    On Error GoTo err1
    SourceFile = Application.InputBox(prompt:="请输入当前文件夹中需要复制文件
    的名称：" & _
        vbCrLf & "（可使用通配符）：", Title:="输入文件名", Type:=2)
    If SourceFile = "Fase" Or SourceFile = "" Then Exit Sub
    If SourceFile = "Fase" Or SourceFile = "" Then Exit Sub
    DestinationFile = Application.InputBox( _
        prompt:="请输入目标文件或文件夹（以“\ ”结尾）的名称：", _
        Title:="目标文件", Type:=2)
    If DestinationFile = "False" Or DestinationFile = "" Then Exit Sub
    SourceFile = ThisWorkbook.Path & "\" & SourceFile  '工作簿当前位置
    fso.CopyFile SourceFile, DestinationFile, True     '复制操作
    MsgBox "复制成功！"
```

```
err1:
    Set fso = Nothing
    If Err.Number <> 0 Then
        MsgBox "复制操作未完成！"
    End If
End Sub
```

以上代码首先让用户输入源文件名（可用通配符表示多个文件），再输入新文件名（或目标文件夹），最后调用 CopyFile 方法完成文件的复制。

23.4.3　分离文件名和扩展名

使用 Application 对象的 GetOpenFilename 方法和 GetSaveAsFilename 方法，可得到一个表示文件位置和名称的字符串，在很多情况下，都需要将文件路径、名称、扩展名分离出来。

使用 VBA 的字符串处理函数可表示对文件名的这 3 部分进行分离，具体代码如下：

```
Function SplitFilename(ByVal sFilename As String) As Variant
    Dim aRet(1 To 3) As String
    Dim i As Integer
    i = InStrRev(sFilename, "\")
    aRet(1) = Left(sFilename, i)
    sFilename = Mid(sFilename, i + 1)
    i = InStrRev(sFilename, ".")
    aRet(2) = Left(sFilename, i - 1)
    aRet(3) = Mid(sFilename, i + 1)
    SplitFilename = aRet
End Function
```

以上函数将文件名全路径名称作为参数，对其进行分离后返回到一个数组中。调用该函数的程序需使用一个数组或 Variant 变量来接收返回值。例如，以下代码可调用 SplitFilename 函数，得到分离的文件名。

```
Sub 分离文件名()
    Dim sFilename As String, aRet As Variant
    sFilename = Application.GetOpenFilename(, , "选择源文件")
    If sFilename = "False" Then Exit Sub    '用户选择"取消"则退出程序
    aRet = SplitFilename(sFilename)
    MsgBox "路径: " & aRet(1) & vbNewLine & _
        "文件名: " & aRet(2) & vbNewLine & _
        "扩展名: " & aRet(3)
End Sub
```

使用 FSO 对象模型提供的方法，可快捷地处理文件名，使用下列方法即得到文件名指定部分的内容。

- GetParentFolderName 方法：该方法根据指定路径中的最后成分返回包含其父文件夹名称的字符串。

- GetBaseName 方法：该方法返回文件的基本名（不带扩展名)，或者提供的路径说明中的文件夹。
- GetExtensionName 方法：使用该方法获取路径最后一个组成部分的扩展名。

下面的代码使用 FSO 对象的以上方法来显示分离的文件名信息。

```
Sub FSO 分离文件名()
    Dim fso As New FileSystemObject, sFileName As String
    Dim str1 As String

    sFileName = Application.GetOpenFilename(, , "选择源文件")
    If sFileName = "False" Then Exit Sub
    MsgBox "路径: " & fso.GetParentFolderName(sFileName) & vbNewLine & _
        "文件名: " & fso.GetBaseName(sFileName) & vbNewLine & _
         "扩展名: " & fso.GetExtensionName(sFileName)
    Set fso = Nothing
End Sub
```

23.5 处理文件夹

很多情况下，在 VBA 中操作文件夹与操作文件的方法类似。例如，复制、删除文件夹等操作。同样，对文件夹的处理也可采用两种方式：传统 VBA 语句和使用 FSO 对象模型的方法。本节介绍常用的文件夹操作方法。

23.5.1 创建文件夹

在 VBA 中可使用 MkDir 语句创建一个新的文件夹。MkDir 语句的语法格式如下：

```
MkDir path
```

其中，参数 path 不能省略，用来指定所要创建的文件夹的字符串。path 可以包含驱动器。如果没有指定驱动器，则 MkDir 会在当前驱动器上创建新的文件夹。

使用 FSO 对象模型中的 CreateFolder 方法也可创建文件夹，该方法的语法格式如下：

```
object.CreateFolder(foldername)
```

其中，参数 foldername 为指明要创建的文件夹。

提示：如果指定的文件夹已经存在，则会出现错误。

以下代码提示用户输入新建文件夹名称，然后调用 FSO 对象的 CreateFolder 方法创建一个文件夹。

```
Sub 新建文件夹()
    Dim fso As New FileSystemObject
    Dim sFolder  As String
```

```
    sFolder = Application.InputBox(prompt:="请输入新建文件夹的名称：", _
        Title:="输入文件夹名称", Type:=2)
    If sFolder = "False" Or sFolder = "" Then Exit Sub
    sFolder = ThisWorkbook.Path & "\" & sFolder
    If fso.FolderExists(sFolder) Then
        MsgBox "文件夹“" & sFolder & "”已经存在！"
    Else
        fso.CreateFolder (sFolder)   '创建文件夹
        MsgBox "文件夹“" & sFolder & "”创建完成！"
    End If
    Set fso = Nothing
End Sub
```

以上代码首先让用户输入新建的文件夹名称，接着判断该文件夹是否存在，若不存在则使用 CreateFolder 方法创建文件夹。

23.5.2　列出文件夹中的文件

在 Windows 的命令模式下，输入命令 Dir，可显示当前文件夹中的文件名称。在 VBA 中，也可列出当前文件夹（或指定文件夹）中包含的文件名。

使用 CurDir 函数可返回指定磁盘的当前的路径。其语法格式如下：

```
CurDir[(drive)]
```

参数 drive 是一个字符串表达式，它指定一个存在的驱动器。如果没有指定驱动器，或 drive 是零长度字符串（""），则 CurDir 会返回当前驱动器的路径。

Dir 函数可返回指定路径下，与指定的模式或文件属性、磁盘卷标相匹配的一个字符串。首先在程序中执行以下语句返回指定路径中的正常文件：

```
Dir(sPath, 0)
```

接着循环执行无参数的 Dir 语句：

```
Dir(sPath, 0)
```

即可获取指定文件夹下的所有文件名。

以下代码使用 Dir 函数列出指定文件夹中的文件：

```
Sub 列出文件夹的文件()
    Dim sPath  As String, sFileName As String, i As Long
    sPath = Application.InputBox(prompt:="请输入文件夹的全路径名称：", _
        Default:=CurDir, Title:="输入文件夹", Type:=2)
    If sPath = "False" Then sPath = CurDir
    sPath = sPath & "\"
    With Sheet1
        .Columns(1).Clear
        .Cells(1, 1) = "“" & sPath & "”文件夹中的文件如下："
        .Cells(1, 1).Font.Bold = True
        sFileName = Dir(sPath, 0)
        i = 1
```

```
        Do While Len(sFileName) > 0
            .Cells(i + 1, 1) = sPath & sFileName
            sFileName = Dir()
            i = i + 1
        Loop
    End With
End Sub
```

以上代码首先让用户输入文件夹名称（默认为工作簿所在文件夹），接着使用 Dir 函数获取指定文件夹中的第一个文件，再使用循环逐个取出指定文件夹下的文件名填充到工作表中。

23.5.3　列出文件夹名称

上节的代码使用 Dir 函数列出指定文件夹中的文件，若在 Dir 函数中可将参数设置为以下形式：

```
Dir(sPath, 16)
```

这样就可列出指定文件夹中包含的子文件夹名称。

要想获得子文件夹更详细的信息，可使用 FSO 对象提供的 Folder 对象来进行操作。使用 GetFolder 方法，根据指定路径中的文件夹返回相应的 Folder 对象。

再使用 Folder 对象的 SubFolders 属性，得到指定文件夹中所有子文件夹（包括隐藏文件夹和系统文件夹）组成的 Folders 集合。对该集合进行循环处理，即可得到所有文件夹的名称、大小等信息。

例如，以下代码可列出 D 盘的文件夹信息：

```
Sub D盘文件夹()
    Dim fso As New Scripting.FileSystemObject
    Dim sFolder As Folder, i As Integer
    On Error Resume Next
    i = 2
    With Worksheets("D盘")
        For Each sFolder In fso.GetFolder("D:\").SubFolders
            .Cells(i, 1) = sFolder.Name
            .Cells(i, 2) = Round(sFolder.Size / 1024, 2) & "KB"
            .Cells(i, 3) = sFolder.ShortName
            i = i + 1
        Next
    End With
    Set fso = Nothing
End Sub
```

23.5.4　删除所有空文件夹

随着磁盘空间的越变越大，磁盘中的文件夹也越来越多。如果要从成千上万个文件夹中查找空文件夹，并将其删除，将是一个非常繁重的工作（有可能不是手工可以完成的任

务）。这时，借助 VBA 代码，通过一定的算法可快速完成该任务。

例如，下面的代码要求用户输入一个文件夹名称，并删除该文件夹及其子文件夹中所有的空文件夹。

```
Sub 删除空文件夹()
    Dim strPath As String
    strPath = Application.InputBox(prompt:="请输入文件夹名称: ", _
        Title:="输入文件夹名称", Default:=ThisWorkbook.Path, Type:=2)
    If strPath = "Fase" Or strPath = "" Then Exit Sub
    Call DelEmtyDir(strPath)
End Sub
```

上面的代码调用 DelEmptyDir 子过程，用来删除指定路径下的空文件夹。该子过程的代码通过递归调用的方法来遍历指定路径下的所有子文件夹，并使用文件夹对象 Folder 的 Size 属性判断文件夹是否为空，如果 Size 属性返回的值为 0，则表示该文件夹为空，调用 Folder 的 Delete 方法将其删除，具体代码如下：

```
Sub DelEmtyDir(ByVal strPath As String)
    Dim fso As New FileSystemObject
    Dim strDirName As String, LastDir As String
    Dim strFld As String, fld As Folder
    If strPath = "Fase" Or strPath = "" Then Exit Sub
    If Right(strPath, 1) <> "\" Then strPath = strPath & "\"
    strDirName = Dir(strPath, vbDirectory)  '取得子文件夹
    Do While strDirName <> ""
                                            '指定文件夹下有子文件夹
        If strDirName <>"." And strDirName <>".."Then  '判断是否为子文件夹
            If (GetAttr(strPath & strDirName) And vbDirectory) = vbDirectory
            Then
                LastDir = strDirName
                Set fld = fso.GetFolder(strPath & strDirName)
                If fld.Size = 0 Then            '文件夹大小为 0,表示为空
                   fld.Delete                   '删除该文件夹
                                                '取得上级文件夹的名称字符串
                   strFld = Left(strPath & strDirName, InStrRev(strPath$ &
                   strDirName, "\") - 1)
                   Call DelEmtyDir(strFld)      '递归调用本过程删除上级文件夹
                Else                            '文件夹不为空
                  Call DelEmtyDir(strPath & strDirName)
                                                '递归调用本过程删除下级子文件夹
                End If
                strDirName = Dir(strPath, vbDirectory)    '处理下一个子文件夹
                Do Until strDirName = LastDir Or strDirName = ""
                    strDirName = Dir
                Loop
                If strDirName = "" Then Exit Do   '如果没有子文件夹,则退出该过程
            End If
        End If
        strDirName = Dir                          '处理下一个子文件夹
    Loop
```

```
    Set fso = Nothing
End Sub
```

以上代码使用了递归调用的方法来遍历指定路径下的所有子文件夹。在递归调用时，必须要有明确条件能退出该子过程，以上代码中通过下面的条件来退出子过程。

```
   If strDirName = "" Then Exit Do
```

23.6　处理文本文件

文本文件是指以 ASCII 码方式（也称文本方式）存储的文件。在文本文件中只能存储英文、数字、汉字等信息，不能存储声音、动画、图像、视频等信息。

23.6.1　创建文本文件

使用 VB 的语句可创建文本文件，也可使用 FSO 对象模型中的相关方法创建文本文件。

1. 使用VB语句创建文本文件

使用 VB 语句操作文件的一般步骤如下：

（1）使用 Open 语句打开文件（使用指定的模式打开文件，即可创建文件）。

（2）对文件进行读写操作。

（3）操作完毕后，关闭文件。

以下代码演示创建文件、写入数据、关闭文件的处理过程：

```
Sub 创建文本文件()
    Dim sFName As String, iFNumber As Integer, r As Long
    sFName = Application.InputBox(prompt:="请输入文本文件的名称：", _
         Title:="输入文件名称", Type:=2)
    If sFName = "False" Or sFName = "" Then Exit Sub
    sFName = ThisWorkbook.Path & "\" & sFName & ".txt"
    iFNumber = FreeFile                         '获取可用文件号
    Open sFName For Output As #iFNumber '用 Output 方式打开文件
    Write #iFNumber, "新建文本文件"             '向文件中写入数据
    Close #iFNumber                             '关闭文件
End Sub
```

2. 使用FSO对象模型的方法创建文本文件

更方便的方法是使用 FSO 对象模型来创建文本文件，常用的方法有以下 3 种：

（1）使用 FileSystemObject 对象的 CreateTextFile 方法。可以使用以下代码创建一个空的文本文件：

```
Dim fso As New FileSystemObject, fil1 As File
```

```
Set fil 1= fso.CreateTextFile(ThisWorkbook.Path & "\test1.txt", True)
```

（2）使用 FileSystemObject 对象的 OpenTextFile 方法，并设置 ForWriting 标志，具体代码如下：

```
Dim fso As New FileSystemObject, ts1 As New TextStream
Set ts1 = fso.OpenTextFile("c:\testfile.txt", ForWriting)
```

（3）使用 File 对象的 OpenAsTextStream 方法，并设置 ForWriting 标志，具体代码如下：

```
Dim fso As New FileSystemObject, fil1 As File, ts As TextStream
Set fil1 = fso.GetFile("C:\Test.txt")
Set ts = fil1.OpenAsTextStream(ForWriting)
```

以下代码使用 FileSystemObject 对象的 CreateTextFile 方法，创建一个文本文件，并向文件中写入一个测试数据。

```
Sub FSO创建文本文件()
    Dim fso As New FileSystemObject
    Dim oStream As TextStream
    Dim sFName As String
    sFName = Application.InputBox(prompt:="请输入文本文件的名称：", _
         Title:="输入文件名称", Type:=2)
    If sFName = "False" Or sFName = "" Then Exit Sub
    sFName = ThisWorkbook.Path & "\" & sFName & ".txt"
                                          '创建文本流对象
    Set oStream = fso.CreateTextFile(Filename:=sFName, OverWrite:=True)
    oStream.WriteLine "新建文本文件"      '向文本流对象中写入数据
    oStream.Close                         '关闭文本流对象
    Set oStream = Nothing
    Set fso = Nothing
End Sub
```

以上代码首先让用户输入文件名，接着用 CreateTextFile 方法创建一个 TextStream 文本流对象，并使用文本流对象的 WriteLine 方法向文件中写入数据。

23.6.2　工作表保存为文本文件

使用 Workbook 对象的 SaveAs 方法，可以将工作表的内容另存为文本文件。例如，下面的代码将工作表 Sheet1 中的数据保存为文本文件：

```
Sub 工作表保存为文本文件()
    Dim sFName As String
    sFName = Application.InputBox(prompt:="请输入文本文件的名称：", _
         Title:="输入文件名称", Type:=2)
    If sFName = "False" Or sFName = "" Then Exit Sub
    sFName = ThisWorkbook.Path & "\" & sFName & ".txt"
    On Error Resume Next
    If Len(Dir(sFName, vbDirectory)) > 0 Then
        If MsgBox("该文件已经存在，是否删除？", vbQuestion + vbYesNo) = vbYes Then
```

```
            Kill sFName     '删除已有的同名文件
        Else
            Exit Sub
        End If
    End If
    On Error GoTo 0
    Set ws1 = Worksheets("Sheet1")
    ActiveWorkbook.SaveAs Filename:=sFName, FileFormat:=xlCSV
    MsgBox "保存成功！"
    ActiveWorkbook.Close SaveChanges:=False
End Sub
```

以上代码首先让用户输入需要保存的文本文件名称，接着检查该文件是否已经存在。如果文件存在，则提示用户是否删除。最后使用 SaveAs 方法将当前工作簿另存为文本文件格式的文件。

23.6.3 添加数据到文本文件

在 FSO 对象模型中，打开文件后就可向文件中添加数据。打开文件的方法有以下两种。

- OpenTextFile 方法：该方法为 FileSystemObject 对象提供的方法，用来打开指定的文件并返回一个 TextStream 对象，可以通过这个对象对文件进行读、写或追加。
- OpenAsTextStream 方法：该方法是 File 对象提供的，用于打开指定的文件并返回一个 TextStream 对象，可以通过这个对象对文件进行读、写或追加。

使用以上两种方法打开文件后，都将返回一个 TextStream 对象，使用该对象的方法可向文本文件中添加数据，常用的方法有以下几种。

- Write 方法：将给定的字符串写入到一个 TextStream 文件中，不换行。
- WriteLine 方法：向 TextStream 文件中写入给定的字符串和一个换行符。
- WriteBlankLines 方法：将指定数量的换行符写入到一个 TextStream 文件中。

例如，以下代码向已有文本文件添加数据：

```
Sub 添加数据到文本文件()
    Dim fso As New FileSystemObject
    Dim oStream As TextStream
    Dim sFName As String
    sFName = Application.GetOpenFilename("文本文件(*.txt),*.txt,所有文件
    (*.*),*.*", _
        1, "打开文本文件")
    If sFName = "False" Then Exit Sub
                                '打开文件为文本流对象
    Set oStream = fso.OpenTextFile(Filename:=sFName, IOMode:=ForAppending)

    With oStream                '追加数据
        .WriteLine "追加测试数据第 1 行！"
        .WriteLine "追加测试数据第 2 行！"
        .WriteLine "追加测试数据第 3 行！"
    End With
```

```
    oStream.Close    '关闭文本流对象
    Set oStream = Nothing
    Set fso = Nothing
End Sub
```

以上代码首先让用户选择要添加数据的文本文件名称，以数据追加方式（ForAppending）打开该文件，并将其引用赋值给一个文本流对象，然后使用 WriteLine 方法向文件中追加数据，最后关闭文本流对象，完成数据的追加操作。

23.6.4　读取文本文件中的数据

与写入数据相对应的，就是从文本文件中读取数据。同样，可使用传统的 VBA 语句从文本文件中获取数据，也可使用 FSO 对象模型提供的方法来获取数据。

1．使用VBA语句获取文件中的数据

可使用 Input #语句、Line Input #语句等从打开的文件中读取数据。在从文件读取数据之前，先要使用 Open 语句打开文件，这时需使用 Input 模式打开文件。

例如，下面的代码从文本文件中读取数据，并填写到当前工作表中：

```
Sub 读取文本文件数据()
    Dim str1 As String, sFName As String, iFNumber As Integer, r As Long
    sFName = Application.GetOpenFilename("文本文件(*.txt),*.txt,所有文件
    (*.*),*.*", _
        1, "打开文本文件")
    If sFName = "False" Then Exit Sub
    iFNumber = FreeFile                    '获取可用文件号
    Open sFName For Input As #iFNumber     '用 Input 方式打开文件
    ActiveSheet.Cells.Clear
    r = 2
    Do
        Line Input #iFNumber, str1
        ActiveSheet.Cells(r, 1) = str1
        r = r + 1
    Loop Until EOF(iFNumber)
    Close #iFNumber                        '关闭文件
End Sub
```

使用 Line Input #语句，一次可从指定的文件中读取一行数据。以上代码通过循环，反复从文本文件中读取数据，直到文件结束为止。使用函数 EOF 可检查是否已经到文件结尾。

2．使用FSO对象模型的方法获取文件数据

使用 FSO 对象模型中的 OpenTextFile 方法或 OpenAsTextStream 方法打开文本文件时，设置 IOMode 参数为 ForReading 模式，这时打开的 TextStream 对象只能读取数据。TextStream 对象提供了以下几个方法读取数据。

- Read 方法：从 TextStream 文件中读取指定数量的字符，并返回由此得到的字符串。

- ReadLine 方法：从 TextStream 文件中读取一整行（一直到换行符，但不包括换行符），并返回由此得到的字符串。
- ReadAll 方法：读取 TextStream 文件的全部内容并返回由此得到的字符串。

以下代码使用 ReadLine 方法从文本文件中逐行读取数据。

```
Sub FSO 读文件数据()
    Dim fso As New FileSystemObject, oStream As TextStream
    Dim sFName As String, str1 As String, r As Long

    sFName = Application.GetOpenFilename("文本文件(*.txt),*.txt,所有文件
    (*.*),*.*", _
        1, "打开文本文件")
    If sFName = "False" Then Exit Sub
                                            '打开文件创建文本流对象
    Set osteam = fso.OpenTextFile(Filename:=sFName, IOMode:=ForReading)
    ActiveSheet.Cells.Clear
    r = 2
    Do
        str1 = oStream.ReadLine
        ActiveSheet.Cells(r, 1) = str1
    Loop Until oStream.AtEndOfStream
    oStream.Close                           '关闭文件
    osteam.Close                            '关闭文件
    Set oStream = Nothing
    Set fso = Nothing
End Sub
```

以上代码使用 TextStream 对象的 AtEndOfStream 属性判断文件是否已到结尾。如果文件指针位于 TextStream 文件末，该属性返回 True；否则返回 False。

第 24 章　使用 ADO 访问数据库

ADO（ActiveX Data Objects）是微软最新的数据访问技术。它被设计用来提供通用数据访问。使用 ADO 可访问各种数据库中的数据，还可访问 Excel 工作簿、文本文件等文件中的数据。

24.1　SQL 结构查询概述

结构化查询语言 SQL（Structured Query Language）是目前各种关系数据库系统广泛采用的标准语言，ADO 对象通过 SQL 语言来操作数据库中的数据。

24.1.1　结构化查询简介

SQL 语言称为“结构化查询语言”，具有 4 部分功能，即查询、操纵、定义和控制。这 4 大功能使 SQL 语言成为通用的、功能强大的关系数据库语言。

SQL 语言的功能强大，但其语法却很简单。完成核心功能一共用了 8 个命令，各命令及功能如下所述。

- ❑ SELECT：用于检索数据，这是使用得最多的一个命令；
- ❑ INSERT：用于增加数据到数据库；
- ❑ UPDATE：用于从数据库中修改现存的数据；
- ❑ DELETE：用于从数据库中删除数据；
- ❑ CREATE：用于创建用户和数据库等；
- ❑ DROP：用于删除表及索引；
- ❑ GRANT：用于对用户进行授权；
- ❑ REVOKE：用于收回用户的授权。

在 Excel 中使用数据库时，主要使用检索数据、插入数据、更新数据、删除数据等功能，下面将简单介绍这些功能。在使用 SQL 语句操作数据库之前，应知道操作的数据库及表结构。本节使用的数据库为“人事管理库”，其中包含表 Emp（保存员工基本数据），该表有 9 列数据，如表 24-1 所示。

表 24-1　表Emp的列

EmpID	员工编号	EmpAge	年龄	EmpIDCard	身份证号
EmpName	姓名	EmpTelephone	电话	EmpEmail	电子邮件地址
EmpSex	性别	EmpAddress	地址	EmpMemo	备注

24.1.2 查询语句 SELECT

SQL 语言的核心是数据库查询语句 SELECT，该语句的功能非常强大，语法格式中的关键字很多，下面列出最常用的一些关键字。更详细的语法描述可参考数据库方面的书籍。

```
SELECT [DISTINCT]  column1 , column2 , ...
FROM tablename1 , tablename2 , ...
[WHERE... ]
[ORDER BY...[ASC | DESC] ]
```

该语句的功能是：从 FROM 子句中给出的表中查询数据，符合 WHERE 子句中设置条件的记录将被筛选出来，被选出的数据只包含 SELECT 子句后面设置的列，并且按 ORDER BY 子句中设置的列进行排序。关键字 ASC 为默认值，表示查询得到的数据按升序排列，若使用 DESC，则表示按降序排列。

最简单的查询语句如下：

```
SELECT *
FROM Emp
```

以上 SQL 语句从名为 Emp 的表中返回所有列和所有记录（即查看整个表的数据）。使用星号（*）表示从表 Emp 中检索数据时包含所有的列。

以下语句将从表 Emp 中返回编号、姓名、性别 3 列的数据，并且返回的数据要满足年龄大于等于 50 这个条件（即查找年龄 50 以上人员的姓名和性别）。

```
SELECT EmpID , EmpName , EmpSex
FROM Emp
WHERE EmpAge>=50
```

查询身份证编号前 3 位数为“101”人员的语句如下：

```
SELECT *
FROM Emp
WHERE EmpIDCard LIKE '101*';
```

查询年龄小于 30 岁的人员，并按姓名排序的语句如下：

```
SELECT *
FROM Emp
WHERE EmpAge<30
ORDER BY EmpName
```

为了增强查询功能，SQL 语言提供了一些内置函数来方便用户查询，这些函数称为库函数，常用的库函数有以下几种。

- ❑ COUNT：对一列中的值计算个数；
- ❑ COUNT(*)：统计满足条件的记录数；
- ❑ SUM：求某一列值的总和（列必须为数值型）；
- ❑ AVG：求某一列值的平均值；

- ❑ MAX：求某一列中的最大值；
- ❑ MIN：求某一列中的最小值。

查询性别为“男”人员的数量，可用以下语句：

```
SELECT COUNT(*)
FROM Emp
WHERE EmpSex= "男"
```

24.1.3　插入语句 INSERT

使用 INSERT 语句，可向数据表中增加记录，该语句的格式如下：

```
INSERT INTO target [(field1[, field2[, ...]])]
VALUES (value1[, value2[, ...]]
```

向 Emp 中增加一个人员资料的语句如下：

```
INSERT INTO Emp(EmpId , EmpName , EmpSex , EmpAge , EmpTelephone , EmpAddress ,
EmpIDCard , EmpEmail , EmpMemo)
VALUES('2008028','吴杰', '男',25,'8010888', '建设路 12 号',
'423123234123456787', 'jueewe@163.com', '')
```

以上语句向 Emp 表的每一列中都增加了数据，对于这种方式插入数据，可省略表名右侧的列表名称，简写为以下样式：

```
INSERT INTO Emp
VALUES('2008028','吴杰', '男',25,'8010888', '建设路 12 号',
'423123234123456787', 'jueewe@163.com', '');
```

在以上语句中，各列的值必须按表中列的顺序给出。

24.1.4　修改语句 UPDATE

修改语句也称为更新语句，可修改表中满足条件记录的对应列的值，其格式如下：

```
UPDATE tablename
SET column1=newvalue1 , column2=newvalue2 , ...
WHERE criteria;
```

修改编号为 2008028 人员的年龄为 30，可使用以下语句：

```
UPDATE Emp
SET EmpAge=30
WHERE EmpID='2008028'
```

24.1.5　删除语句 DELETE

删除语句用来删除表中满足指定条件的记录，其格式如下：

```
DELETE FROM tablename
WHERE criteria
```

DELETE 语句删除指定表中满足条件的记录。

例如，删除表 Emp 中年龄大于 60 的记录，其语句如下：

```
DELETE FROM Emp
WHERE EmpAge>60
```

注意：如果省略 WHERE 子句，则将指定表中的所有记录都删除！

24.2　ADO 对象模型

ADO（ActiveX Data Objects）是微软最新的数据访问技术。其主要优点是易于使用、高速度、低内存支出和占用磁盘空间较少。

24.2.1　ADO 对象模型

在 VBA 中使用 ADO 对象，必须先为前工程引用 ADO 的对象库，可单击主菜单【工具】|【引用】命令，打开【引用】对话框，在列表框中找到 Microsoft ActiveX Data Objects 2.8 Library 选项并选中，如图 24-1 所示。

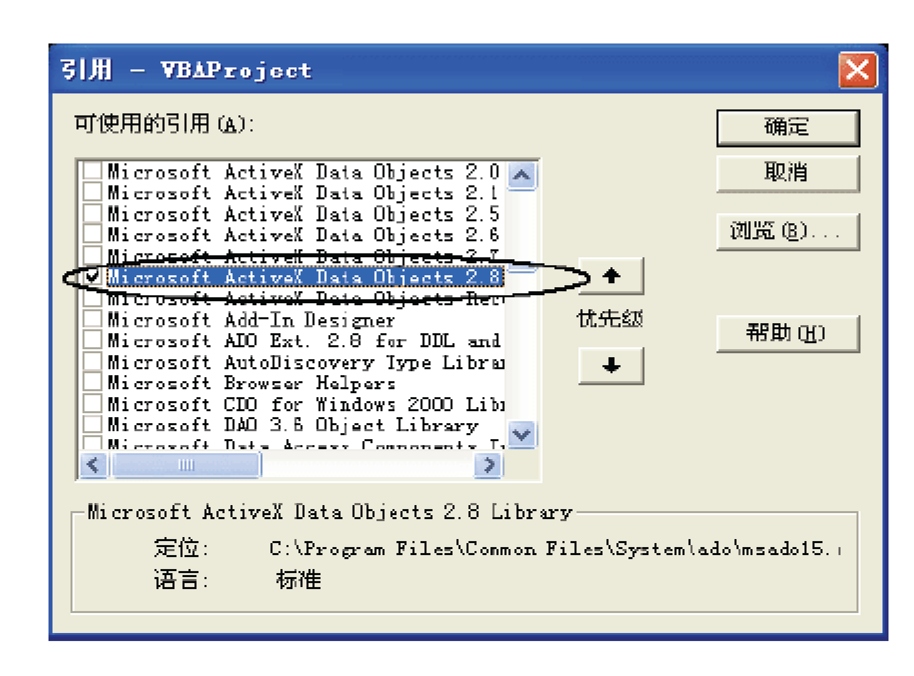


图 24-1　引用 ADO 对象

ADO 对象模型定义了一个可编程的分层对象集合，主要由三个对象成员（Connection、Command 和 Recordset）和集合对象 Errors、Parameters 及 Fields 等组成，如图 24-2 所示。

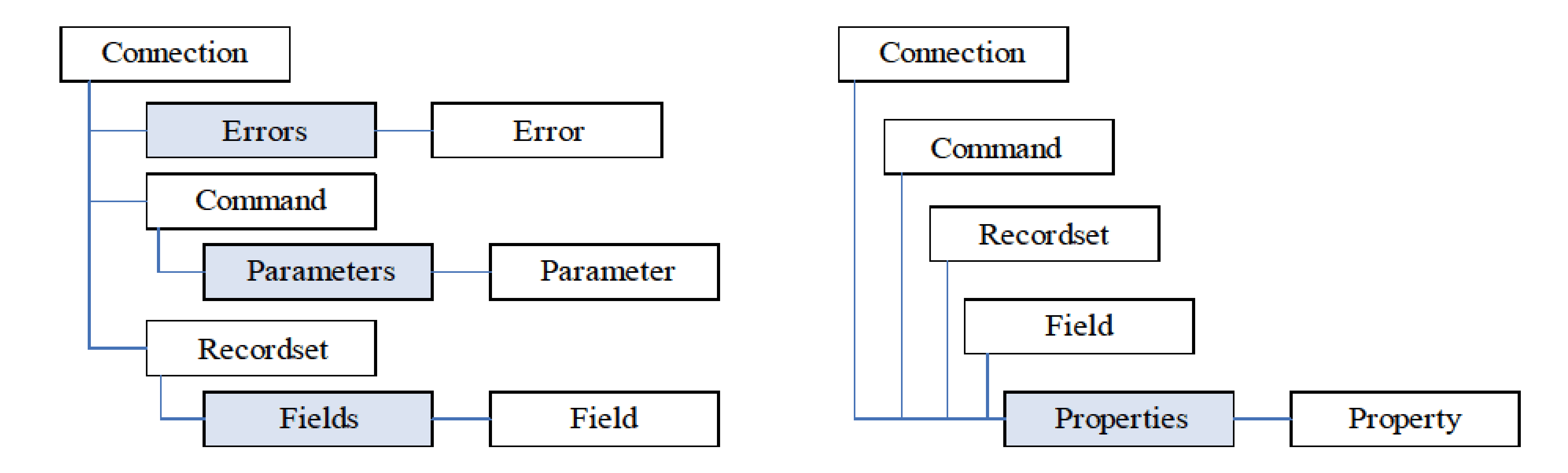


图 24-2　ADO 对象模型

24.2.2　Connection 对象

Connection 对象表示数据源的连接，通过该对象可生成 ADO 层次中的其他对象，其返回值是一个 Recordset 对象。

Connection 对象提供属性和方法对连接进行控制。

1．设置连接字符串

ConnectionString 属性是 Connection 对象最常用的属性，设置用于建立连接数据源的信息。通过传递包含一系列由分号分隔的 argument = value 语句的详细连接字符串可指定数据源。连接到不同的数据库时其连接字符串也不同。

以下连接字符串连接到名为“人事管理.mdb”的 Access 数据库：

```
Connection.ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=人事管理.mdb"
```

也可使用以下字符串连接：

```
Connection.ConnectionString="Driver={Microsoft  Access  Driver  (*.mdb)};
Dbq=人事管理.mdb; Uid=Admin;Pwd=;"
```

2．打开连接

给 Connection 对象设置好各种属性后，还需要使用 Open 方法来打开数据源的连接，该方法的语法格式如下：

```
connection.Open ConnectionString, UserID, Password, Options
```

该方法可接收以下 4 个参数，这些参数都可省略。

- ConnectionString：包含连接信息的字符串，若在 ConnectionString 属性中已设置了连接字符串，就不需要在此处重新设置了。
- UserID：建立连接时所使用的用户名。
- Password：建立连接时所使用的密码。
- Options：设置该方法是同步打开连接还是异步打开连接。可设置为常量 adConnectUnspecified（同步）和 adAsyncConnect（异步打开连接）。

在对打开的 Connection 的操作结束后，可使用 Close 方法释放所有关联的系统资源。关闭对象并非将它从内存中删除，可以更改它的属性设置并在以后再次使用 Open 方法打开它。要将对象完全从内存中删除，可将对象变量设置为 Nothing。

3．执行SQL语句

使用 Connection 对象的 Execute 方法可执行指定的查询、SQL 语句、存储过程或特定提供者的文本等内容。其语法格式如下：

```
Set recordset = connection.Execute (CommandText, RecordsAffected, Options)
```

Execute 方法执行后，将结构返回到一个 Recordset 对象中，方便用户处理返回的数据。Execute 方法使用下面的 3 个参数。

- CommandText：包含要执行的 SQL 语句、表名、存储过程或特定提供者的文本。
- RecordsAffected：提供者向其返回操作所影响的记录数目，该参数可省略。
- Options：指示提供者应如何计算 CommandText 参数，可设置为如表 24-2 所示的常量。

表 24-2　Options常量

常　量	说　明
AdCmdText	指示提供者应按命令的文本定义计算 CommandText
AdCmdTable	指示 ADO 应生成 SQL 查询，以便从 CommandText 命名的表中返回所有行
AdCmdTableDirect	指示提供者应从 CommandText 命名的表中返回所有行
AdCmdTable	指示提供者应按表名计算 CommandText
AdCmdStoredProc	指示提供者应按存储过程计算 CommandText
AdCmdUnknown	指示 CommandText 参数中的命令类型未知
adAsyncExecute	指示命令应该异步执行
adAsyncFetch	指示对在 CacheSize 属性指定的初始数量之后的剩余行使用异步提取

注意：使用 Connection 对象的 Execute 方法，返回的 Recordset 对象始终为只读、仅向前的游标。如需要具有更多功能的 Recordset 对象，应首先创建具有所需属性设置的 Recordset 对象，然后使用 Recordset 对象的 Open 方法执行查询并返回所需的游标类型。

24.2.3　Recordset 对象

Recordset 对象表示的是来自基本表或命令执行结果的记录全集。该对象所指的当前记录均为集合内的单个记录。可使用该对象操作来自提供者的数据。使用 ADO 时，通过该对象几乎可对所有数据进行操作。所有 Recordset 对象均使用记录（行）和字段（列）进行构造。

任何时候，Recordset 对象所指的当前记录均为集合内的单个记录。

1．获取记录集

有两种方式可获得记录集，一种是用 Connection 对象的 Execute 方法，另一种是使用 Recordset 对象的 Open 方法，其语法格式如下：

```
recordset.Open Source, ActiveConnection, CursorType, LockType, Options
```

各参数的含义如下所述。

- Source：为变量名、SQL 语句、表名、存储过程调用等。
- ActiveConnection：为连接数据库的连接字符串。
- CursorType：确定提供者打开 Recordset 时应该使用的游标类型。

- LockType：确定提供者打开 Recordset 时应该使用的锁定（并发）类型。
- Options：用于指示提供者如何计算 Source 参数（如果它代表的不是 Command 对象），或从以前保存 Recordset 的文件中恢复 Recordset。

2. 增加记录

如果需要向记录集中新增记录，可以使用 AddNew 方法。其语法格式为：

```
recordset.AddNew FieldList, Values
```

两个参数的含义如下所述。

- FieldList：设置新记录中字段的单个、一组字段名称或序列位置。
- Values：为新记录中字段的单个或一组值。如果 Fields 是数组，那么 Values 也必须是有相同成员数的数组，否则将发生错误。字段名称的次序必须与每个数组中的字段值的次序相匹配。

如果在编辑当前记录或添加新记录时调用 AddNew 方法，ADO 将调用 Update 方法保存原有记录的更改再创建新记录。

3. 删除记录

如果要删除当前记录或记录组，可使用 Delete 方法。其语法格式为：

```
recordset.Delete AffectRecords
```

参数 AffectRecords 确定 Delete 方法所影响的记录数目。该值有两种可能：一是 adAffectCurrent，表示仅删除当前记录，为默认值；另一个是 adAffectGroup，表示删除满足当前 Filter 属性设置的记录。使用该选项须将 Filter 属性设置为有效的值。

4. 更新记录

如果已经对当前记录集进行了修改或添加了新的记录，则需要更新（保存），可以使用 Update 方法更新记录集。CancelUpdate 方法为取消修改。

5. 访问特定字段的数据

在 Recordset 对象中有一个集合对象 Fields，其中包含了各字段的名称和数据。通过如下语法格式可以访问特定字段的值：

```
recordset.Fields(index)[.Value | Name]
```

参数 index 是变体型，可以是要访问的字段名或字段在记录集中的顺序号。Value 和 Name 是字段的属性，分别表示字段的值和字段的名称。默认为 Value。

6. 移动记录集指针

记录集对象有 4 个方法用来移动记录指针，它们是 MoveFirst、MovePrevious、MoveNext 和 MoveLast，分别可以移动到第一条记录、上一条记录、下一条记录和最后一条记录。其语法格式为：

```
Recordset.MoveFirst
```

如果需要大跨度的移动记录指针，则需使用 Move 方法，它可直接移动到指定的记录。其语法格式为：

```
Recordset.Move n [,s]
```

其中，n 是个数值表达式，表示相对当前记录移动多少条记录，如果为正数则向表的后方移动，如果为负数则向表前方移动。s 是可选项，指明移动的起始位置，该参数可取 3 个值，adBookmarkCurrent 是默认值，表示从当前记录开始；adBookmarkFirst 表示从第一条记录开始；adBookmarkLast 表示从最后一条记录开始。

7．取得记录集的总记录数

Recordset 的属性 RecordCount 返回 Recordset 对象中记录的当前数目。

8．记录集的顶和底

在移动记录指针时，如果到记录集的顶部时还执行 MovePrevious 方法，就会发生错误。同样，到记录集的尾部时使用 MoveNext 方法也会出错。所以在使用移动方法之前，一般都要先判断当前记录的位置，可用 BOF 和 EOF 两个属性来判断。

- BOF：指示当前记录位置位于 Recordset 对象的第一个记录之前。
- EOF：指示当前记录位置位于 Recordset 对象的最后一个记录之后。

如果当前记录位于第一个记录之前，BOF 属性将返回 True；如果当前记录为第一个记录或位于其后，则将返回 False。

如果当前记录位于 Recordset 对象的最后一个记录之后，EOF 属性将返回 True；如果当前记录为 Recordset 对象的最后一个记录或位于其前，则将返回 False。

24.2.4 其他 ADO 常用对象

前面介绍了 Connection 对象、Recordset 对象的常用属性和方法。ADO 对象模型还提供多个对象，下面列出这些对象的作用。

- Command 对象：该对象定义了将对数据源执行的命令。使用 Command 对象可查询数据库并返回 Recordset 对象中的记录，以便执行大量操作或处理数据库结构。
- Error 对象：访问数据源时所返回的错误信息。每个错误出现时，一个或多个 Error 对象将被放到 Connection 对象的 Errors 集合中。当另一个 ADO 操作产生错误时，Errors 集合将被清空，并在其中放入新的 Error 对象集。
- Parameter 对象：与命令对象有关的参数。
- Field 对象：该对象表示 Recordset 对象中的一个数据列（注意不是一行），Recordset 对象含有由 Field 对象组成的 Fields 集合，每个 Field 对象对应于 Recordset 中的一列。使用 Field 对象的 Value 属性可设置或返回当前记录的数据。

24.2.5　使用 ADO 访问数据库的步骤

使用 ADO 对象编程访问数据库，一般按以下步骤编写代码。

（1）使用 Connection 对象连接到数据源；

（2）使用 Recordset 对象的打开记录集方法获得记录集对象（也可使用 Command 对象执行 SQL 语句获得记录集对象）；

（3）在程序中访问记录集中的数据（添加、更新、删除等操作）；

（4）使用 Connection 对象中断连接。

例如：执行以下代码，在“人事管理”数据库的表 Emp 中进行查询，将 EmpID 列为 2008028 人员的资料显示到工作表中。

```
Sub 员工资料()
    Dim cnn As New Connection, rst As Recordset, fld As Field
    Dim strSql As String, i As Long, strConn As String
    strSql = "SELECT * FROM Emp WHERE EmpID='2008028'"
    strConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source='"
    strConn = strConn & ActiveWorkbook.Path & "\人事管理.mdb'"
    cnn.ConnectionString = strConn      '设置连接字符串
    cnn.Open '打开连接
    Set rst = cnn.Execute(strSql)       '执行 SQL 语句生成 Recordset
    With Worksheets("sheet1")
        i = 1
        For Each fld In rst.Fields      '输出表头
            .Cells(1, i) = fld.Name     '每列为一个字段名
            i = i + 1
        Next
        j = 1
        Do While Not rst.EOF            '循环处理记录集中的记录本
            i = 1
            j = j + 1
            For Each fld In rst.Fields  '循环处理各字段
                .Cells(j, i) = fld.Value '显示字段的值
                i = i + 1
            Next
            rst.MoveNext '下一记录
        Loop
    End With
    Set rst = Nothing
    Set cnn = Nothing
End Sub
```

24.3　访问 Excel 工作簿的数据

因为 Excel 具有易用性、通用性和庞大的用户群，所以在一些小的应用程序中，可以将 Excel 作为后台数据库，用来保存用户的数据。本节实例演示用 ADO 方式访问 Excel 数

据库的方法。

24.3.1 查询工作表中的数据

使用 ADO 方式访问 Excel 工作簿，就是将 Excel 工作簿作为一个数据库，工作簿中每个工作表为一个数据表，工作表中的一列数据为一个字段。

用 ADO 访问 Excel 工作簿，需使用类似下面的连接字符串：

```
"Provider=Microsoft.Jet.OLEDB.4.0; Extended Properties=Excel 8.0; _
   Data Source=工作簿名称"
```

其中 Extended Properties 设置访问工作簿的版本号。

例如，在当前工作簿的工作表“员工”中，保存着员工资料。在工作表 Sheet2 的单元格 B2 中输入姓名，单击右侧的【查询】按钮，即可使用 ADO 方式从工作表“员工”中查找数据。在查找数据时，使用 Like 关键字进行模糊查询（即输入姓名中的某一个或几个字符，即可查询出）。具体的代码如下：

```
Sub 按姓名查询()
    Dim cnn As New Connection, rs As New Recordset
    Dim strSql As String, str1 As String
    On Error Resume Next
    cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" _
           & "Extended Properties=Excel 8.0;" _
           & "Data Source=" & ThisWorkbook.FullName
    str1 = Worksheets("sheet2").Range("B2")
    strSql = "Select * FROM [员工$] Where 姓名 like '%" & str1 & "%'"
    rs.Open strSql, cnn, adOpenStatic
    With Worksheets("sheet2")
        .Range("A4:I100").ClearContents
        .Range("A4").CopyFromRecordset rs
    End With
    rs.Close
    cnn.Close
    Set rs = Nothing
    Set cnn = Nothing
End Sub
```

以上代码首先创建数据库连接，再根据用户输入的字符组成一个 SQL 语句，用该 SQL 语句打开记录集，最后使用 CopyFromRecordset 方法将记录集中的数据复制到工作表中。

使用 CopyFromRecordset 方法可将 Recordset 对象中的内容复制到工作表，从指定区域的左上角开始。该方法的语法格式如下：

```
表达式.CopyFromRecordset(Data, MaxRows, MaxColumns)
```

各参数的含义如下所述。

- ❑ Data：复制到区域的 Recordset 对象。
- ❑ MaxRows：复制到工作表上的最大记录数。如果省略该参数，将复制 Recordset 对

象中的所有记录。

- MaxColumns：复制到工作表上的最大字段数。如果省略该参数，将复制 Recordset 对象中的所有字段。

提示：复制从 Recordset 对象的当前行开始。复制完成后，Recordset 对象的 EOF 属性为 True。

24.3.2　导入其他工作表数据

使用 SQL 的 INSERT 语句可向表中添加一个或多个记录。下面的代码使用 INSERT 语句将另一工作簿中的数据添加到当前工作表中。

```
Sub 导入数据()
    Dim cnn As New Connection
    Dim strSql As String, str1 As String
    On Error Resume Next
    cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" _
            & "Extended Properties=Excel 8.0;" _
            & "Data Source=" & ThisWorkbook.FullName

    strSql = "INSERT INTO [员工$A:I] SELECT * FROM  "
    strSql = strSql & " [Excel 8.0;Database="
    strSql = strSql & ThisWorkbook.Path & "\使用 ADO.xls" & ";HDR=YES].[员
    工$A:I];"
    cnn.Execute strSql

    cnn.Close
    Set cnn = Nothing
End Sub
```

以上代码首先创建当前工作簿的 ADO 连接，接着创建 SQL 语句从原工作簿（此处为“使用 ADO.xls”文件）的指定工作表（“员工”工作表）中选择数据，并插入到目标工作簿中。

在 Excel 工作簿中，每个工作表相当于数据库中的表，每列为一个字段，使用符号$分隔表和列。使用下面的语句从工作簿“使用 ADO.xls”的工作表“员工”中获取 A～I 列的数据：

```
SELECT * FROM  [Excel 8.0;Database=使用 ADO.xls;HDR=YES].[员工$A:I];
```

其中的 HDR=YES 表示工作表有表头（第 1 行为字段名）。

使用下面的语句即可将指定工作表中的数据增加到当前工作簿中：

```
INSERT INTO [员工$A:I] SELECT * FROM [Excel 8.0;Database=使用
ADO.xls;HDR=YES].[员工$A:I];
```

提示：使用 ADO 方式，可以在不打开工作簿的情况下获取工作簿中的数据。

24.4 访问 Access 数据库

将 Excel 作为数据库可使用在一些小型应用中，对于需要处理大量数据时，更多的还是使用专业的数据库系统（如 Access、SQL Server 数据库系统等）。在很多情况下，用户希望从这些专业的数据库系统中获取部分数据，然后在 Excel 中进行分析处理。

下面介绍在 Excel 中操作 Access 数据库的方法，包括从 Access 数据库中导入数据到 Excel、将 Excel 工作表中的数据添加到 Access、修改数据、删除数据等操作。

本节使用 Access 系统中提供的 Northwind.mdb 作为示例数据库。

24.4.1 导入 Access 数据

将 Access 数据库中的数据导入 Excel 的操作比较简单，主要有以下 3 个步骤：

（1）首先使用 ADO 连接到数据库。

（2）根据需要，设置不同的查询条件创建查询记录集。

（3）将记录集中的数据复制到指定工作表中。

（4）关闭数据库连接。

以下代码从示例数据库中导入“客户”表中的所有数据：

```
Sub 获取客户信息()
    Dim cnn As New Connection, rs As New Recordset
    Dim strSql As String, i As Long, sh As Worksheet

    On Error Resume Next
    cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" _
           & "Data Source=" & ThisWorkbook.Path & "\Northwind.mdb"
    strSql = "Select * FROM [客户]"           '从“客户”表中获取全部数据
    rs.Open strSql, cnn, adOpenStatic         '打开记录集
    Set sh = Worksheets.Add                   '添加工作表
    sh.Name = "客户信息"                       '设置工作表名称
    With sh
        For i = 0 To rs.Fields.Count - 1      '用字段名作为表头
            .Cells(1, i + 1) = rs.Fields(i).Name
        Next
        .Range("A2").CopyFromRecordset rs     '复制记录集中的数据
        .Columns.AutoFit                      '设置列宽为自动适应
    End With
    rs.Close
    cnn.Close
    Set rs = Nothing
    Set cnn = Nothing
End Sub
```

以上代码首先创建 Northwind 数据库的连接，接着创建 SQL 语句从数据库中查询获取

数据，最后将每个字段的名称填入到工作表的第 1 行作为表头，并使用 CopyFromRecordset 方法将记录集中的所有记录填充到当前工作表中。

24.4.2　添加数据到 Access

使用 INSERT INTO 语句可将 Excel 工作表中的数据添加到 Access 数据库的指定表中。这时可使用 VBA 代码读取工作表中的数据，并生成一个 INSERT INTO 语句，然后再调用 Connection 对象的 Execute 方法执行该 SQL 语句，即可完成添加数据的操作。

为了使 Excel 工作表中数据的输入与数据库指定表的字段保持对应关系，可首先打开数据库中的表，获取字段名填充到表中作为表头。如图 24-3 所示，单击【生成表单】按钮即可将 Northwind 数据库中的“供应商”表的字段填到 A 列中。该按钮的 VBA 代码如下：

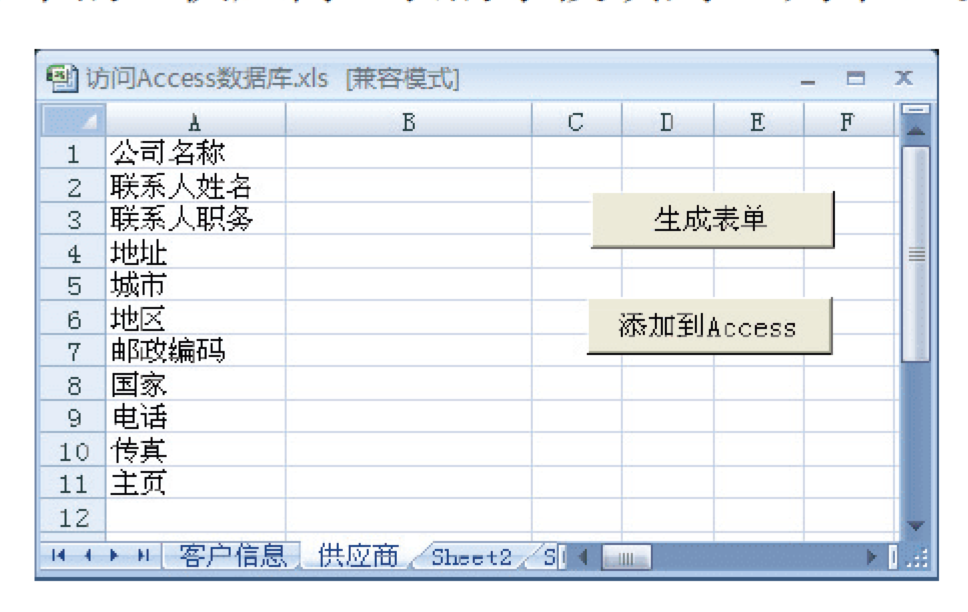


图 24-3　添加数据到 Access

```
Sub 生成表单()
    Dim cnn As New Connection, rs As New Recordset, fd As Field
    Dim strSql As String, i As Long
    On Error Resume Next
    cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" _
            & "Data Source=" & ThisWorkbook.Path & "\Northwind.mdb"
    rs.Open "select * from 供应商", cnn
    With Worksheets("供应商")
        For i = 1 To rs.Fields.Count
            .Cells(i, 1) = rs.Fields(i).Name
        Next
    End With
    cnn.Close
    Set cnn = Nothing
    Set rs = Nothing
End Sub
```

提示：“供应商”表中的第 1 个字段由系统自动生成，不需要用户输入，因此不显示在图 24-3 所示的表格中。

用户在图 24-3 所示的表单中输入各字段的数据后，单击【添加到 Access】按钮，即可将输入的数据添加数据库的“供应商”表的最后。具体的代码如下：

```
Sub 添加供应商信息()
    Dim cnn As New Connection
    Dim strSql As String, i As Long, c As Long

    On Error Resume Next
    cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" _
            & "Data Source=" & ThisWorkbook.Path & "\Northwind.mdb"
    strSql = "INSERT INTO 供应商("
    With Worksheets("供应商")
        c = .Range("A1").End(xlDown).Row      '数据行数
        For i = 1 To c                         '生成字段列表
            strSql = strSql & .Cells(i, 1) & ","
        Next
        strSql = Left(strSql, Len(strSql) - 1) & ") VALUES("
        For i = 1 To c                         '生成插入的值列表
            strSql = strSql & "'" & .Cells(i, 2) & "',"
        Next
        strSql = Left(strSql, Len(strSql) - 1) & ")"
    End With
    cnn.Execute strSql                         '执行 SQL 语句
    cnn.Close
    Set cnn = Nothing
End Sub
```

以上代码通过工作表中的数据生成 INSERT INTO 语句。首先从 A 列获取“供应商”表的字段名，生成 INSERT 语句的部分信息。接着用 B 列的数据生成 VALUES 部分的数据。

注意：生成 VALUES 数据时，必须使用逗号将这些值分隔，并且将文本字段用引号(' ')括起来。

24.4.3 修改记录

使用 UPDATE 语句可对数据库中的记录进行修改。在使用 UPDATE 语句时，一般都需要使用 WHERE 子句限制需要修改记录的范围，如果不使用 WHERE 子句，则表示表中的相应字段都将被修改成相同的数据。

例如，要修改 Northwind 数据库“客户”表中的“公司名称”，可使用以下代码：

```
Sub 修改客户名称()
    Dim cnn As New Connection, strcon As String
    Dim strSql As String, custID As String, custName As String

    With Worksheets("修改")
        custID = .Range("B1")     '获取客户 ID
        custName = .Range("B2")   '获取公司名称
    End With
    If Trim(custID) = "" Or Trim(custName) = "" Then
```

```
        MsgBox "请输入“客户 ID”和“公司名称”信息！", vbCritical + vbOKOnly
        Exit Sub
    End If

    On Error Resume Next
    strcon = "Provider=Microsoft.Jet.OLEDB.4.0;" _
        & "Data Source=" & ThisWorkbook.Path & "\Northwind.mdb"
    cnn.Open strcon                                 '打开数据库连接
    strSql = "UPDATE 客户 SET 公司名称='" & custName & _
        "' WHERE 客户 ID='" & custID & "'"         '修改"客户"表的 SQL 语句
    cnn.Execute (strSql)                            '执行 SQL 命令
    cnn.Close
    Set cnn = Nothing
End Sub
```

以上代码首先检查用户在工作表中输入的内容，再根据工作表中的内容生成 UPDATE 语句，最后执行该 SQL 语句完成数据的修改操作。

以上代码在 WHERE 子句中使用主键作为条件，每次只修改一条记录。在更多的情况下，表中满足 WHERE 子句设置条件的记录有多个，这些记录的数据都将被修改。例如，以下代码将地区为东北的客户名称前面加上记录对应的地区名称：

```
UPDATE 客户 SET 公司名称='(' +地区+')'+公司名称 where 地区='东北'
```

24.4.4　删除记录

使用 SQL 的 DELETE 语句来删除数据表中的记录。与 UPDATE 语句类似，在定义 DELETE 语句时，如果不指定 WHERE 子句，将删除表中所有记录。这是相当危险的一个操作。

还可以用 Execute 方法和 DROP 语句从数据库中删除整个表。不过，如果用这种方法删除表，将会失去表的结构。不同的是当使用 DELETE 删除表时，只有数据会被删除，表的结构以及表的所有属性仍然保留，例如字段属性及索引。

例如，以下代码可删除满足指定条件的记录：

```
Sub 删除记录()
    Dim cnn As New Connection
    Dim strcon As String, strSql As String, custID As String
    custID = Worksheets("修改").Range("B1")
    If Trim(custID) = "" Then
        MsgBox "请输入“客户 ID”信息！", vbCritical + vbOKOnly
        Exit Sub
    End If
    On Error Resume Next
    strcon = "Provider=Microsoft.Jet.OLEDB.4.0;" _
        & "Data Source=" & ThisWorkbook.Path & "\Northwind.mdb"
    cnn.Open strcon                                                '打开数据库连接
    strSql = "DELETE FROM 客户 WHERE 客户 ID='"&custID & "'"  '删除 SQL 语句
    cnn.Execute (strSql)
```

```
    If Err <> 0 Then                                          '显示错误信息
        MsgBox Err.Description
    End If
    cnn.Close
    Set cnn = Nothing
End Sub
```

以上代码从修改工作表的“B1”单元格获取要删除的客户 ID，再使用 DELETE 语句删除该条记录。

如果将 SQL 语句定义为如下：

```
strSql = "DELETE FROM 客户"
```

执行该 SQL 语句将删除“客户”表中的全部记录。

24.4.5 创建 Access 数据库

在 VBA 中可以通过编程的方式创建一个新的 Access 数据库。要使用 ADO 创建数据库，除了要引用对象库 Microsoft ActiveX Data Objects 2.8 Library 之外，还需要引用另一个对象库 Microsoft ADO Ext. 2.8 For DDL Security，如图 24-4 所示。

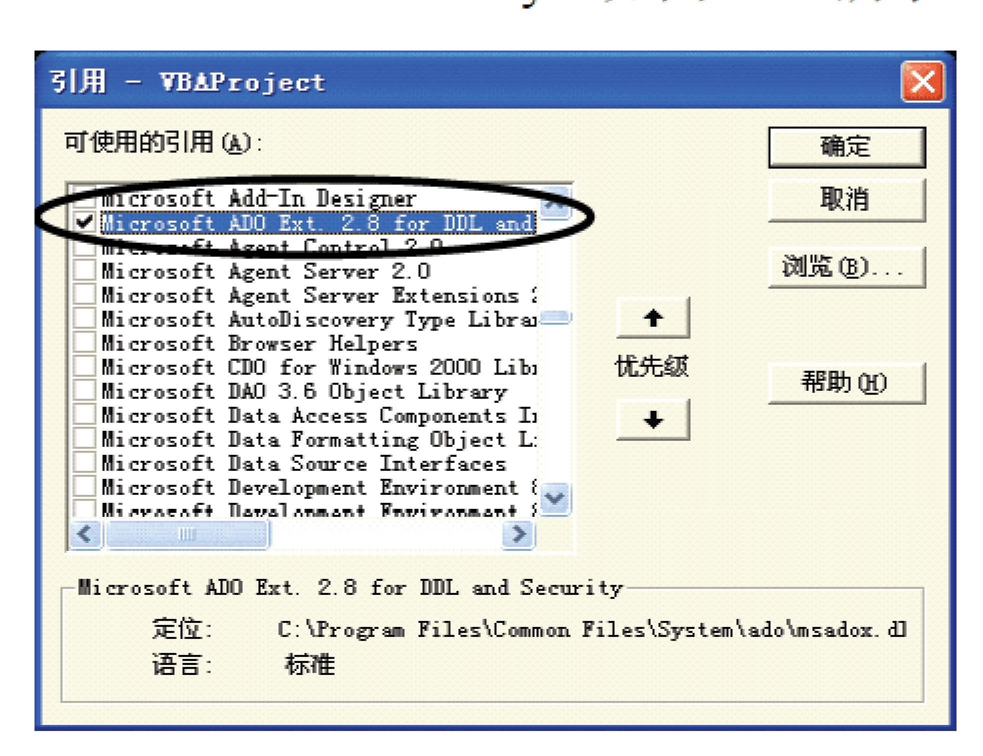


图 24-4 引用 ADO Ext 对象库

在工程中引用 Microsoft ADO Ext. 2.8 For DDL Security 对象库后，即可使用 ADOX. Catalog 对象来创建 Access 数据库。

以下代码可在 Excel 中创建新的 Access 数据库。

```
Sub 创建数据库()
    Dim cat As New ADOX.Catalog, rs1 As New Recordset
    Dim conn As New Connection, str1 As String, strSql As String
    Dim sDBName As String

    sDBName = Application.GetSaveAsFilename("新建数据库", _
        "Access 数据库(*.mdb),*.mdb", 1, "输入数据库名称")
    If sDBName = "False" Then Exit Sub
    str1 = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & sDBName
```

```
    strSql = "Create Table test(学号 Char(10),姓名 Char(10),性别 Char(2)," & _
        " 出生日期 Date,联系电话 Char(20),地址 Char(40))"
    cat.Create str1                                                '创建数据库
    conn.Open str1                                                 '打开新建的数据库
    conn.Execute strSql                                            '创建新表
    With rs1
        .Open "test", conn, adOpenKeyset, adLockPessimistic '创建记录集
        .AddNew  '添加记录
        .Fields("学号") = "Y0001"                                  '为各字段赋值
        .Fields("姓名") = "张新"
        .Fields("性别") = "男"
        .Fields("出生日期") = #1/1/1988#
        .Fields("联系电话") = "3311778"
        .Fields("地址") = "建华巷 58 号"
        .Update
    End With
    rs1.Close
    conn.Close
    Set conn = Nothing
End Sub
```

以上代码首先使用 ADOX.Catalog 对象创建一个 Access 数据库，再使用 Create 语句在新建数据库中创建一个表，最后向表中添加一个测试数据。

24.4.6　列出所有表名

ADOX 对象库中提供了多个对象，在 24.2.5 节中节使用了 Catalog 对象创建新的数据。使用 Table 对象可检索数据库中包含的表名及其他属性。Table 对象表示数据库表，包括列、索引和关键字。可使用 Table 对象的以下属性和集合。

- ❑ Name 属性：表示数据表的名称。
- ❑ Type 属性：表示数据表的类型。
- ❑ Columns 集合：访问表的数据库列。
- ❑ Indexes 集合：访问表的索引。
- ❑ Keys 集合：访问表的关键字。
- ❑ ParentCatalog 属性：指定拥有表的 Catalog。
- ❑ DateCreated 和 DateModified 属性：返回日期信息。
- ❑ Properties 集合：访问特定提供者的表属性。

使用以下代码可显示 Northwind 数据库中的表名称：

```
Sub 显示表名()
    Dim cat As New ADOX.Catalog , strcon As String, r As Long
    strcon = "Provider=Microsoft.Jet.OLEDB.4.0;" _
        & "Data Source=" & ThisWorkbook.Path & "\Northwind.mdb"
    cat.ActiveConnection = strcon
    r = 3                                   '工作表中显示数据的行
```

```
    With Worksheets("表名")
        .Columns(1).Clear
        .Cells(2, 1) = "表: "
        .Cells(2, 1).Font.Bold = True           '加粗字体
        For i = 0 To cat.Tables.Count - 1       '列出表名称
            If cat.Tables(i).Type = "TABLE" Then '类型为“TABLE”就是表
                .Cells(r, 1) = cat.Tables(i).Name '表名称
                r = r + 1
            End If
        Next
   End With
    Set cat.ActiveConnection = Nothing
End Sub
```

24.4.7　表的字段信息

Recordset 对象含有由 Field 对象组成的 Fields 集合。每个 Field 对象对应于 Recordset 中的一列（字段）。使用 Field 对象的 Value 属性可设置或返回当前记录的数据。而使用 Fields 对象的属性还可获取字段的名称、类型等信息。

Field 对象的常用集合、方法、和属性分别如下所述。

- Name 属性：表示字段名。
- Value 属性：表示字段中的数据。
- Type 属性：表示字段类型
- Precision 属性：表示数字字段的精度。
- NumericScale 属性：表示数字字段值的范围。
- DefinedSize 属性：表示字段大小。
- ActualSize 属性：表示字段的值的实际长度。
- AppendChunk 方法：将数据追加到大型文本、二进制数据字段中。
- GetChunk 方法：返回大型文本或二进制数据字段对象的全部或部分内容。

表的字段信息包含在 Fields 对象中，通过 VBA 代码循环处理 Fields 对象，即可获取表的字段信息。例如，以下代码在 Excel 工作表“字段信息”中列出了数据库表“客户”的字段信息。

```
Sub 表的字段信息()
    Dim cnn As New Connection, rs As New Recordset
    Dim strcon As String, strSql As String, r As Long
    On Error Resume Next
    strcon = "Provider=Microsoft.Jet.OLEDB.4.0;" _
        & "Data Source=" & ThisWorkbook.Path & "\Northwind.mdb"
    cnn.Open strcon                                     '打开数据源连接
    strSql = "客户"                                     'SQL 语句
    rs.Open strSql, cnn, adOpenKeyset, adLockOptimistic
    With Worksheets("字段信息")
        .Cells.Clear                                    '清除表中的数据
        .Range("A1:C1") = Array("名称", "类型", "大小")    '填写表头数据
```

```
        .Range("A1:C1").Font.Bold = True
        For r = 0 To rs.Fields.Count - 1                    '将字段信息填充到工作表
            .Cells(r + 2, 1) = rs.Fields(r).Name            '字段名
            .Cells(r + 2, 2) = rs.Fields(r).Type            '字段类型
            .Cells(r + 2, 3) = rs.Fields(r).DefinedSize     '字段大小
        Next
    End With
    rs.Close
    cnn.Close
    Set rs = Nothing
    Set cnn = Nothing
End Sub
```

以上代码将“客户”表中的数据装入记录集中，再循环处理 Fields 集合对象中的每个 Field 对象（字段），通过 Field 对象的属性即可得到表的字段信息。

使用 Recordset 对象的 Open 方法时，可直接给出一个表的名称，这样将在记录集中返回该表中的所有数据，与下面的 SELECT 语句功能相同：

```
SELECT * FROM 客户
```

第 25 章　Excel 2007 与 Internet

随着互联网的普及，Office 软件对 Internet 的支持也越来越强大。使用 Excel 不但可以浏览访问 Internet，还可以直接将电子表格发布为网页。

25.1　管理超链接

超链接是指从一个网页指向一个目标的链接关系，这个目标可以是另一个网页，也可以是相同网页上的不同位置，还可以是一个图片、一个电子邮件地址、一个文件，甚至是一个应用程序。而用来超链接的对象，可以是一段文本或者是一个图片。

25.1.1　插入超链接

在 Excel 操作界面中，可以方便地向工作表中插入超链接，以链接到指定的文档、网页等不同目标。在 Excel 中插入超链接的步骤如下：

（1）在工作表上，单击选择要创建超链接的单元格（或工作表中的图像）。

（2）在【插入】选项卡上的【链接】组中，单击【超链接】按钮，打开如图 25-1 所示【插入超链接】对话框。

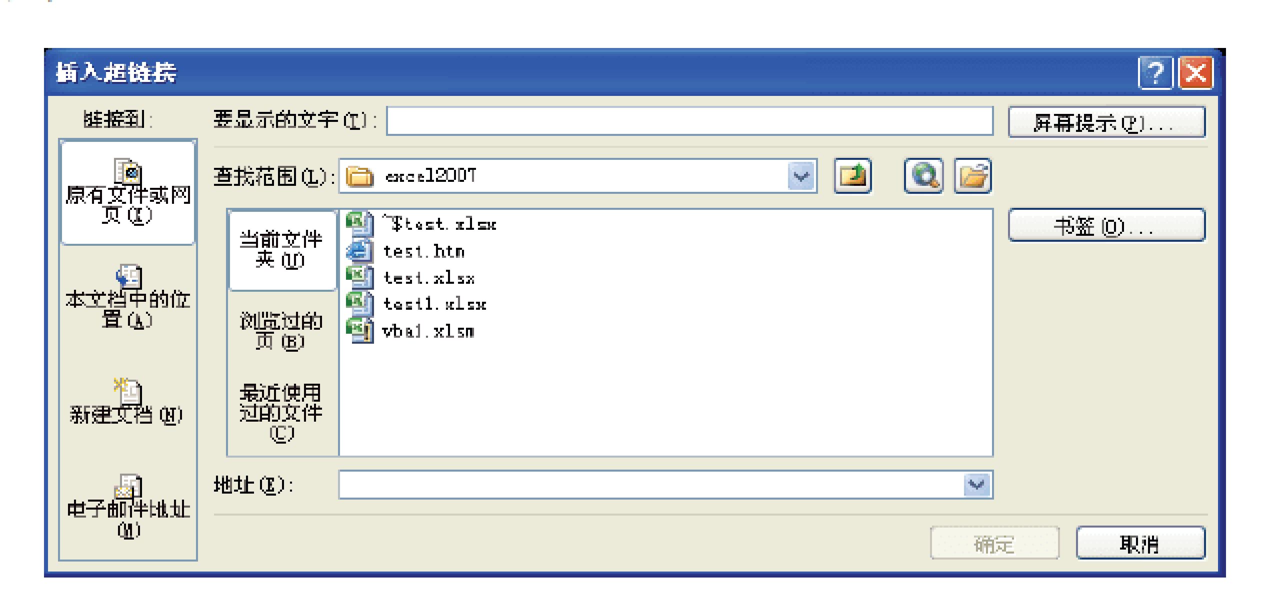


图 25-1　【插入超链接】对话框

（3）在对话框左边的【链接到】选项中选择链接目标的类型，在右侧输入链接的地址、文件名等信息，单击【确定】按钮，即可为选择的单元格（或图像）插入超链接。

25.1.2　用 VBA 创建超链接

除了在Excel界面中手工插入超链接，还可通过编写VBA代码向工作表中插入超链接。

在 VBA 中，使用 Hyperlink 对象表示超链接，所有超链接组成 Hyperlinks 集合对象。

使用 Hyperlinks 集合对象的 Add 方法，可向指定的区域或形状添加超链接。其语法格式如下：

```
表达式.Add(Anchor, Address, SubAddress, ScreenTip, TextToDisplay)
```

各参数的含义如下所述。

- Anchor：设置插入超链接的位置。可为 Range 或 Shape 对象。
- Address：超链接的地址，可以是一个文件或网页地址。
- SubAddress：超链接的子地址，该参数可省略。
- ScreenTip：当鼠标指针停留在超链接上时所显示的屏幕提示，该参数可省略。
- TextToDisplay：要显示的超链接的文本，该参数可省略。

例如，使用以下代码，将在工作表的单元格 B1 中添加一个超链接，设置超链接的目标为百度网首页。

```
Sub 链接到网页()
    Dim rng As Range, hlk As Hyperlink
    Set rng = Worksheets("sheet1").Range("B1")
    Set hlk = rng.Hyperlinks.Add(anchor:=rng, Address:="http://www.baidu.
    com")
    With hlk
        .Address = "http://www.baidu.com"
        .ScreenTip = "百度"
        .TextToDisplay = "浏览百度网站"
    End With
End Sub
```

执行以上代码，将在工作表 Sheet1 的单元格 B1 中添加一个超链接，如图 25-2 所示。单击该超链接，即可在 IE 浏览器中打开百度网站。



图 25-2　插入超链接

25.1.3　添加超链接到收藏夹

收藏夹是一个网页收藏工具，可以将看到的网址收藏起来，方便以后查看。Hyperlink 对象的 AddToFavorites 方法可以将工作簿或超链接的快捷方式添加到“收藏夹”文件夹中。

例如，以下代码将工作表 Sheet1 单元格 B3 中的网址添加到收藏夹中。

```
Sub 添加到收藏夹()
```

```
    Dim hlk As Hyperlink, rng As Range
    Set rng = Worksheets("sheet1").Range("b3")
    Set hlk = rng.Hyperlinks.Add(anchor:=rng, Address:=rng.Value)
    hlk.AddToFavorites
End Sub
```

在如图 25-3 所示工作表的单元格 B3 中输入网址，单击右侧的【添加到收藏夹】按钮，即可将其添加到 IE 的收藏夹中。打开 IE 浏览器，在下拉菜单【收藏】中可看到添加的网址。

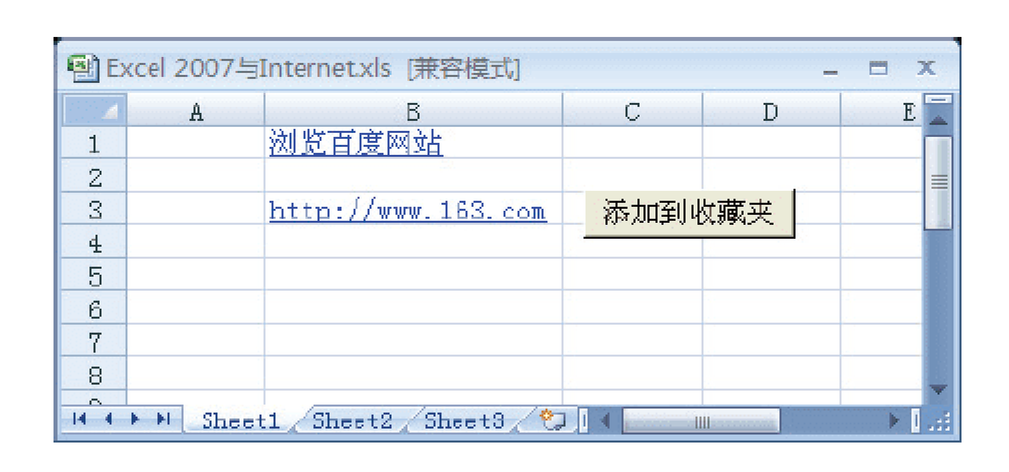


图 25-3　添加到收藏夹

25.1.4　直接打开网页

使用 Hyperlink 对象调用网页时，必须指定 Anchor 参数，即指定工作表中的单元格或图像作为锚点。这样，工作表中将显示超链接的文字。

若要求在一定条件下打开指定网页，又不希望在工作表中显示超链接文字，则可使用 Workbook 对象的 FollowHyperlink 方法。该方法对指定超链接进行处理以下载目标文档，然后将该文档在适当的应用程序中显示出来。该方法语法格式如下：

```
表 达 式 .FollowHyperlink(Address, SubAddress, NewWindow, AddHistory, ExtraInfo, Method, HeaderInfo)
```

各参数的含义如下所述。

- ❑ Address：表示目标文档的地址。
- ❑ SubAddress：表示目标文档中的位置。默认值为空字符串。
- ❑ NewWindow：如果为 True，则在新窗口中显示目标应用程序。默认值为 False。
- ❑ AddHistory：未使用。保留供将来使用。
- ❑ ExtraInfo：指定解析超链接时要使用的 HTTP 附加信息。例如，可以指定查询字符串等。
- ❑ Method：指定附加 ExtraInfo 的方法。可以是常量 msoMethodGet 或 msoMethodPost 之一。
- ❑ HeaderInfo：指定 HTTP 请求的标题信息的 String。默认值为空字符串。

例如，编写代码完成以下功能，在如图 25-4 所示工作表的单元格 B5 中输入关键字，单击右侧的【搜索】按钮，将打开百度网站搜索该关键字，如图 25-5 所示。

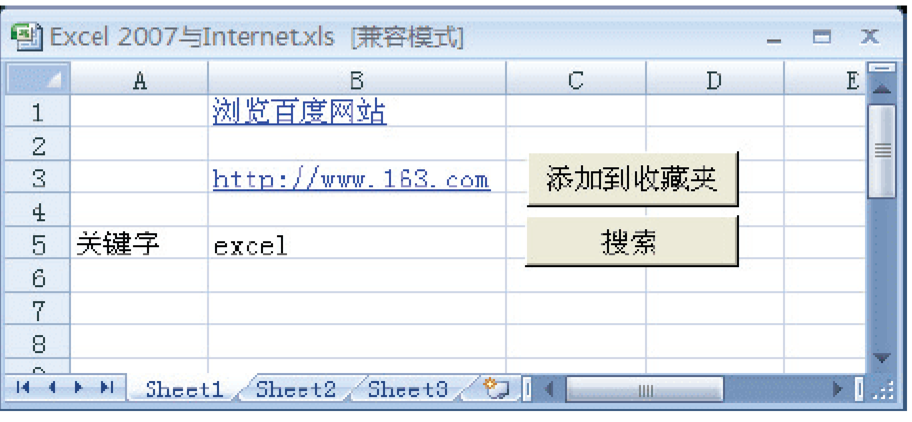


图 25-4　输入搜索关键字



图 25-5　搜索

具体的代码如下：

```
Sub 搜索()
    Dim str1 As String
    str1 = Worksheets("sheet1").Range("B5").Value
    If str1 <> "" Then
        ActiveWorkbook.FollowHyperlink Address:="http://www.baidu.com/s", _
            Extrainfo:="wd=" & str1, _
            Method:=msoMethodGet
    End If
End Sub
```

以上代码中，设置参数 Method 为 msoMethodGet，即将 Extrainfo 参数中的内容添加到 Address 后面，得到如下所示的网址字符串（设 Extranfo 的参数为“wd=excel”）：

```
http://www.baidu.com/s?wd=excel
```

在以上网址字符串中，问号（？）由系统自动添加，表示该符号后的字符串为提交给网页的参数。有关网页参数提交方面的知识，请读者参阅 HTML 相关书籍。

25.2　打开 Internet 上的工作簿

使用 Excel 可以打开本地或远程 Web 服务器上的工作簿，还可以打开 HTML 格式的网页文档，也可以将工作簿保存到 Web 服务器中。

25.2.1　打开 Web 上的工作簿

对于已经连接到 Internet 的用户，就可使用 Excel 的【打开】对话框打开指定的网站，

或网站中保存的 Excel 工作簿。具体步骤如下：

（1）单击【Office 按钮】打开下拉菜单。

（2）在下拉菜单中选择【打开】命令，将打开如图 25-6 所示的【打开】对话框。

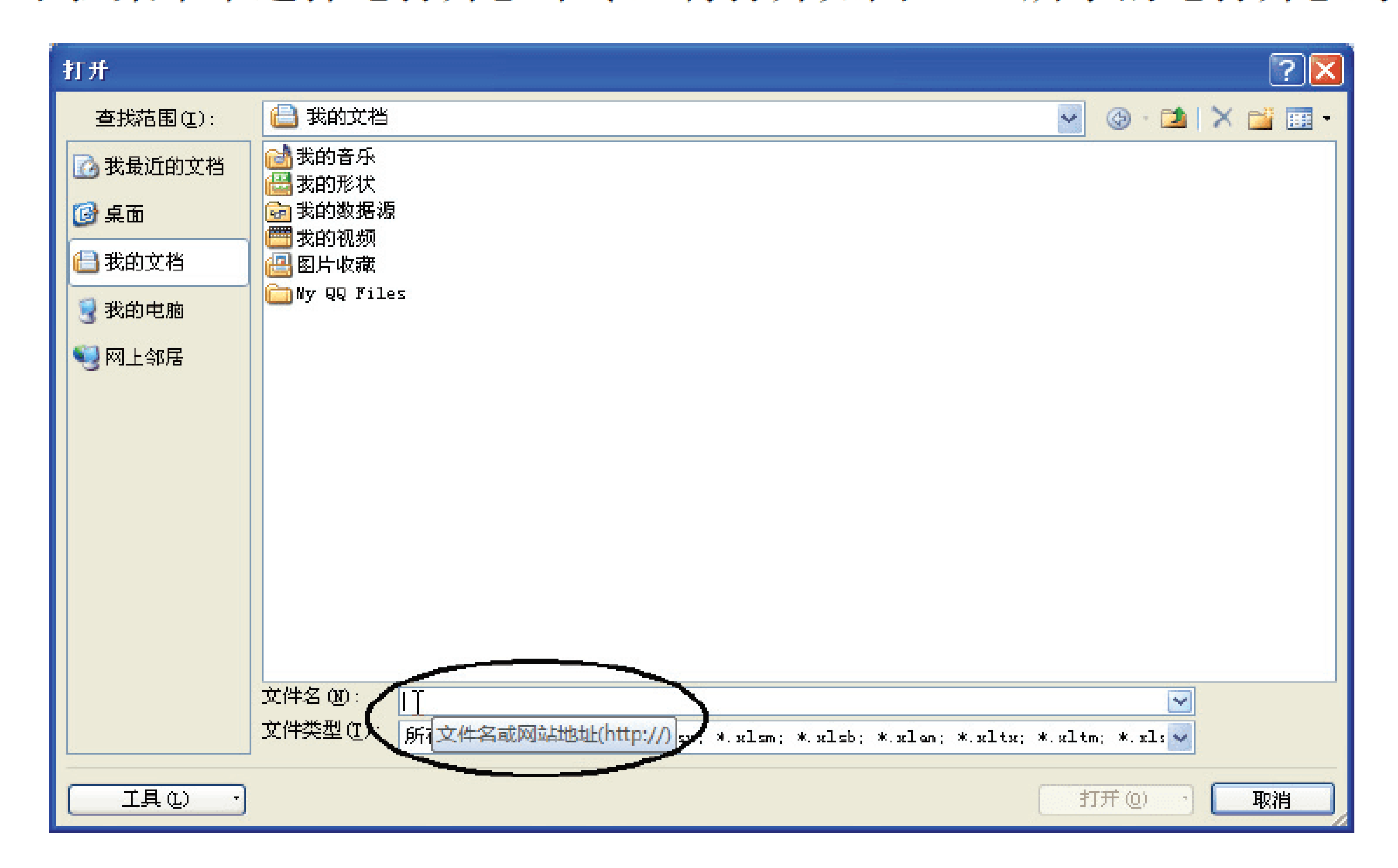


图 25-6 【打开】对话框

（3）在文件名文本框中输入网站地址后，按 Enter 键，将出现图 25-7 所示的对话框，提示正在连接到网站。连接成功后，将在 Excel 窗口中打开网页。



图 25-7 连接网络

提示：要将工作簿保存到 Web 服务器，在【另存为】对话框的【文件名】文本框中输入网站域名和工作簿名称即可。

25.2.2 用 VBA 代码打开 Web 上的工作簿

使用 Workbooks 集合对象的 Open 方法，可打开本地工作簿，也可打开指定 Web 服务器中的工作簿。例如，以下代码打开保存在服务器中的工作簿 http://www.test.com/test.xls。

```
Workbooks.Open("http://www.test.com/test.xls")
```

同样，要将工作簿保存到 Web 服务器中，可使用下面的语句：

```
ActiveWorkbook.SaveAs("http://www.test.com/test.xls")
```

提示：要将工作簿保存到 Web 服务器中，必须在服务器上运行 FrontPage Server Extensions。

25.3　使用 Internet 上的数据

将网页中的数据导入 Excel 的方法有很多种，最简单的方法就是选定网页表格内容，执行复制操作，在 Excel 中粘贴就可以了。但网页上有些数据是时常更新的，使用这种方法，Excel 工作表中的数据不会随网站一起更新。通过创建 Web 查询，可从网页上更新数据。

25.3.1　创建 Web 查询

Excel 支持创建可自动更新的 Web 查询。在 Excel 中创建 Web 查询的步骤如下：

（1）在【数据】选项卡的【获取外部数据】组中，单击【自网站】命令按钮，打开如图 25-8 所示的【新建 Web 查询】对话框。

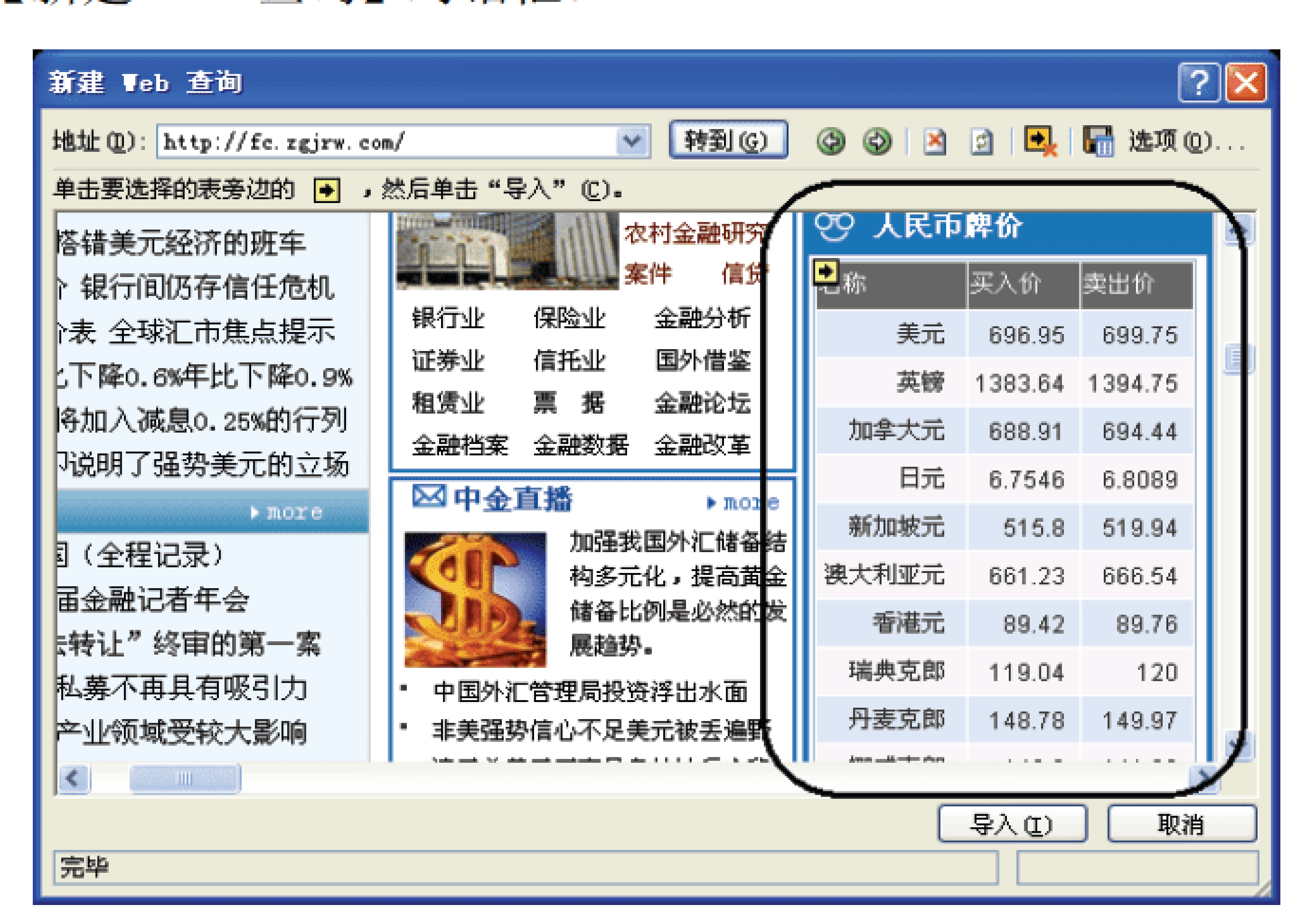


图 25-8 【新建 Web 查询】对话框

（2）在对话框的【地址】下拉列表中输入网站地址（本例输入 http://fc.zgjrw.com/），单击右侧的【转到】按钮，将在对话框下方打开该网站。

（3）拖动滚动条，在浏览窗口中找到“人民币牌价”。

（4）从图 25-8 中可以看到有表格的地方其左上角都会出现一个黄色的小箭头，单击该箭头就可以使其变成小勾，表示该箭头所指的表格将被导入 Excel 中。单击“人民币牌价”左侧的黑色小箭头，将该部分表格的数据导入 Excel 中。

（5）全部选定后可以单击对话框右上方的【保存查询】按钮，将该 Web 查询单独保存，这样可在以后其他工作簿中重复调用。

（6）单击对话框右下方的【导入】按钮，打开【导入数据】对话框，如图 25-9 所示，设置将导入的数据保存在当前工作表中，还是新建一个工作表。

（7）单击【确定】按钮后，一个可根据网页数据更新的表格创建完毕，如图 25-10 所示。

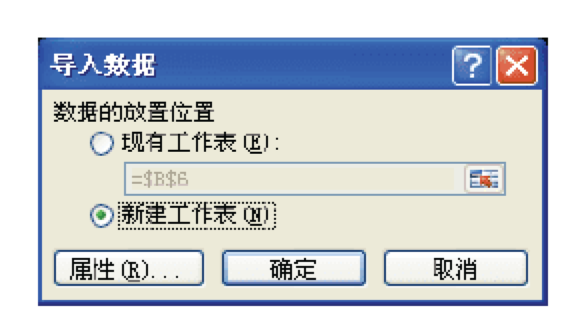


图 25-9　导入数据


Excel 2007与Internet.xls [兼容模式]

	A	B	C	D
1	名称	买入价	卖出价	
2	美元	696.95	699.75	
3	英镑	1383.64	1394.75	
4	加拿大元	688.91	694.44	
5	日元	6.7546	6.8089	
6	新加坡元	515.8	519.94	
7	澳大利亚元	661.23	666.54	
8	香港元	89.42	89.76	
9	瑞典克郎	119.04	120	
10	丹麦克郎	148.78	149.97	
11	挪威克郎	140.2	141.33	
12	瑞士法郎	692.23	697.79	
13	欧元	1110.11	1119.02	

Sheet4　Sheet1　Shel

图 25-10　导入的数据

25.3.2　了解 QueryTable 对象

在 Excel 界面中，通过【新建 Web 查询】对话框可创建 Web 查询，也可使用 VBA 代码创建 Web 查询。每一个 Web 查询就是一个 QueryTable 对象。

QueryTable 对象代表一个利用外部数据源（如 SQL Server、Access 数据库、网络数据）返回的数据生成的工作表表格。QueryTable 对象是 QueryTables 集合的成员。

1．Add方法

使用 QueryTables 集合对象的 Add 方法可新建一个查询表。其语法格式如下：

```
表达式.Add(Connection, Destination, Sql)
```

该方法参数的含义如下：

- ❑ Connection：可以是一个 Web 查询。Web 查询字符串的格式如下：

```
URL; http://fc.zgjrw.com/
```

- ❑ Destination：查询表目标区域（生成的查询表的放置区域）左上角的单元格。目标区域必须位于 QueryTables 对象所在的工作表中。
- ❑ Sql：在 ODBC 数据源上运行的 SQL 查询字符串。当使用的数据源为 ODBC 数据源时，在 Web 查询中该参数省略。

2．Refresh方法

使用 QueryTable 对象的 Refresh 方法可更新外部数据区域(QueryTable)。该方法的语法格式如下：

```
表达式.Refresh(BackgroundQuery)
```

参数 BackgroundQuery 如果为 True，则在数据库建立连接并提交查询之后，将控制返回给过程。QueryTable 在后台进行更新。如果为 False，则在所有数据被取回到工作表之后，

将控制返回给过程。如果没有指定该参数，则由 BackgroundQuery 的属性设置决定查询模式。

在 Excel 建立一个成功的连接之后，将存储完整的连接字符串，这样，以后在同一编辑会话中调用 Refresh 方法时就不会再显示提示。通过检查 Connection 属性的值可以获得完整的连接字符串。

如果成功地完成或启动查询，则 Refresh 方法返回 True；如果用户取消连接或参数对话框，该方法返回 False。

25.3.3　用 VBA 创建 Web 查询

了解 QueryTable 对象的方法后，可使用以下 VBA 代码创建图 25-8 所示的 Web 查询：

```
Sub VBA 创建 Web 查询()
    Dim qt As QueryTable, ws As Worksheet
    Set ws = Worksheets.Add
    Set qt = ws.QueryTables.Add( _
        Connection:="URL;http://info.stockstar.com/partner/zgjr/rmbpj.
        asp", _
        Destination:=ws.Range("A1"))
    qt.Refresh
End Sub
```

在 QueryTables 集合对象的 Add 方法中，设置 Connection 参数时，必须以“URL;”开头，再加上需要获取数据的网址。

25.3.4　带参数的 Web 查询

在 25.3.3 节的例子中，通过 VBA 代码创建的 Web 查询获取网页中所有的数据。在更多的情况下，需要根据用户输入的值从网站中查询结果。这时，需要设置带参数的 Web 参数。

在网站设计中，向网页提交参数有两种方法：GET 和 POST。

- GET 方式通过 URL 提交数据，提交后在地址栏中的地址类似以下形式：

```
http://www.baidu.com/s?wd=abc
```

其中问号（？）后的字符串就是提交的数据。

- POST 方式在向网页提交数据时，不是通过 URL 提交数据，必须另外通过变量提交参数。使用这种方式提交数据时，地址栏中的地址看不出变化。

1. 提交GET方式的参数

在 VBA 中，编写带参数的 Web 查询时，若网站设计要求提交 GET 方式的参数，只需要获取用户输入的参数，然后将其附加在 URL 地址后，再提交即可得到需要的数据。

很多快递公司都在公司的网站上提供了查询快件投递情况的服务。下面以申通快递为

例，介绍设计带参数 Web 查询的方法。

要设计动态查询，首先应了解快递公司提供查询页面的 URL，然后输入一个编号进行测试查询，再观察 IE 浏览器地址栏中参数的名称，最后将这些数据应用到 VBA 程序中。具体的步骤如下：

（1）打开 IE 浏览器，打开快递公司网站主页，在左侧查询框中输入快件编号，如图 25-11 所示。



图 25-11　在 IE 中查询

（2）单击【查询】按钮，打开新的 IE 窗口，在地址栏中可看到类似下面的 URL：

```
http://www.sto.cn/querybill/webform1.aspx?wen=268480394601
&Submit2=%B2%E9%D1%AF
```

以上 URL 表示使用网页“http://www.sto.cn/querybill/webform1.aspx”提供查询功能，快件单号保存在参数“wen”中。

（3）取得以上参数后，就可以方便地设置带参数的 Web 查询，具体代码如下：

```
Sub 查询快件()
    Dim str As String, strURL As String

    str = Application.InputBox(prompt:="请输入快件的编号：", _
        Title:="申通快件查询", Type:=2)
    If str = "False" Then Exit Sub
    strURL = "URL;http://www.sto.cn/querybill/webform1.aspx?wen="
    strURL = strURL & str & "&Submit2=%E6%9F%A5%E8%AF%A2"
    With Worksheets.Add '新增工作表
        .Name = str       '设置快件编号为工作表名
    End With
```

```
    With ActiveSheet.QueryTables.Add(Connection:=strURL, Destination:=
    Range("A2"))
        .Name = "sto"
        .FieldNames = True
        .WebSelectionType = xlSpecifiedTables     '导入指定表
        .WebFormatting = xlWebFormattingNone      '不导入任何格式
        .WebTables = "1,2"                        '导入第一个和第二个表格中的数据
        .BackgroundQuery = True                   '查询异步执行（在后台执行）
        .Refresh                                  '更新数据
    End With
End Sub
```

执行以上代码，弹出如图 25-12 所示对话框，让用户输入查询快件的单号。

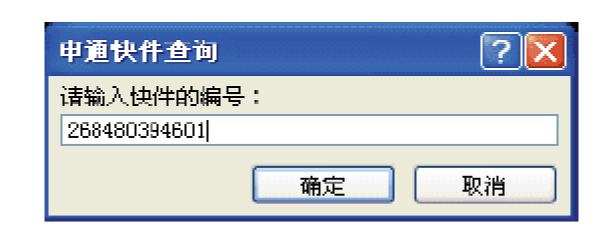


图 25-12　输入编号

在图 25-12 所示对话框中输入快件编号，单击【确定】按钮进行查询，将查询结果添加到新增的工作表中，如图 25-13 所示。

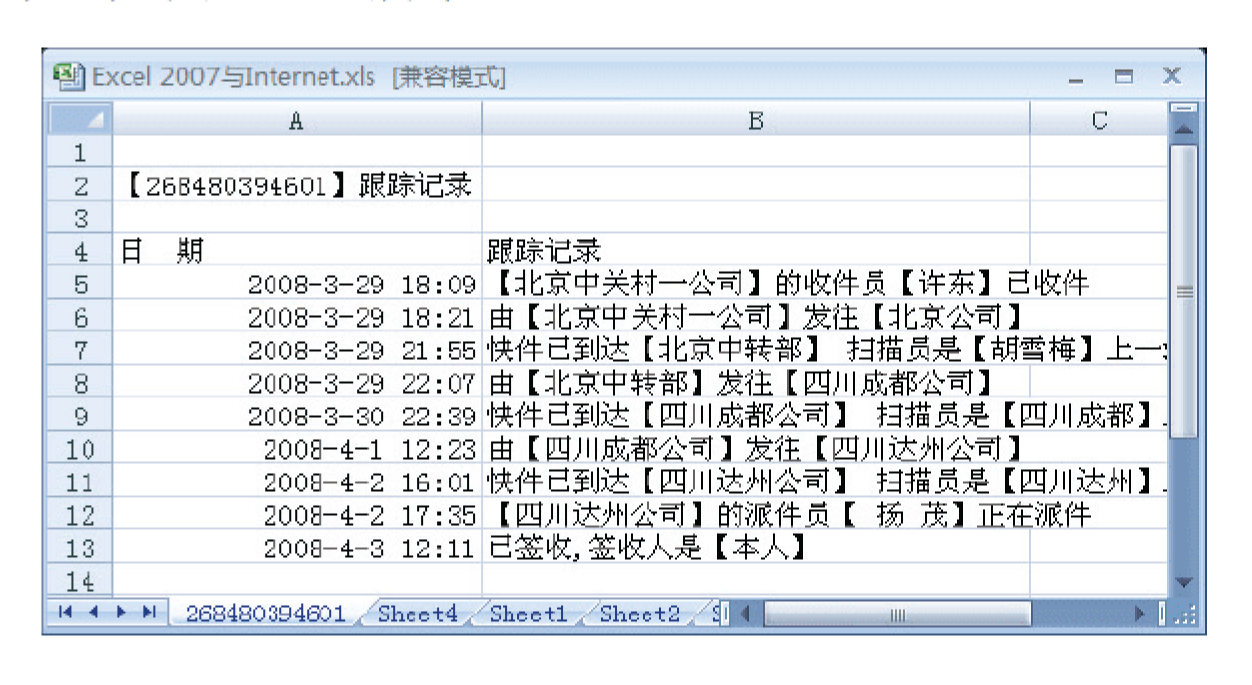


图 25-13　查询结果

注意：在使用以上代码进行查询之前，计算机应该能访问互联网。

2．提交POST方式的参数

与 GET 方式提交参数不同，使用 POST 方式提交参数时，在 URL 地址栏看不到提示。这时，需要将参数通过 QueryTable 对象的 PostText 属性进行提交。PostText 属性返回或设置用于 POST 方法的字符串。

使用 POST 方式提交参数时，要知道网页中需要提交参数的名称也比较麻烦（有一定网页设计经验的读者应该能简单地查找各参数名）。下面用实例介绍操作过程。

很多网站都提供查询手机号码所在地的服务，下面演示用 VBA 设置 Web 查询查找手机号码所在地方法。

（1）在 IE 浏览器中输入地址 http://www.ip138.com/，打开如图 25-14 所示的网页。

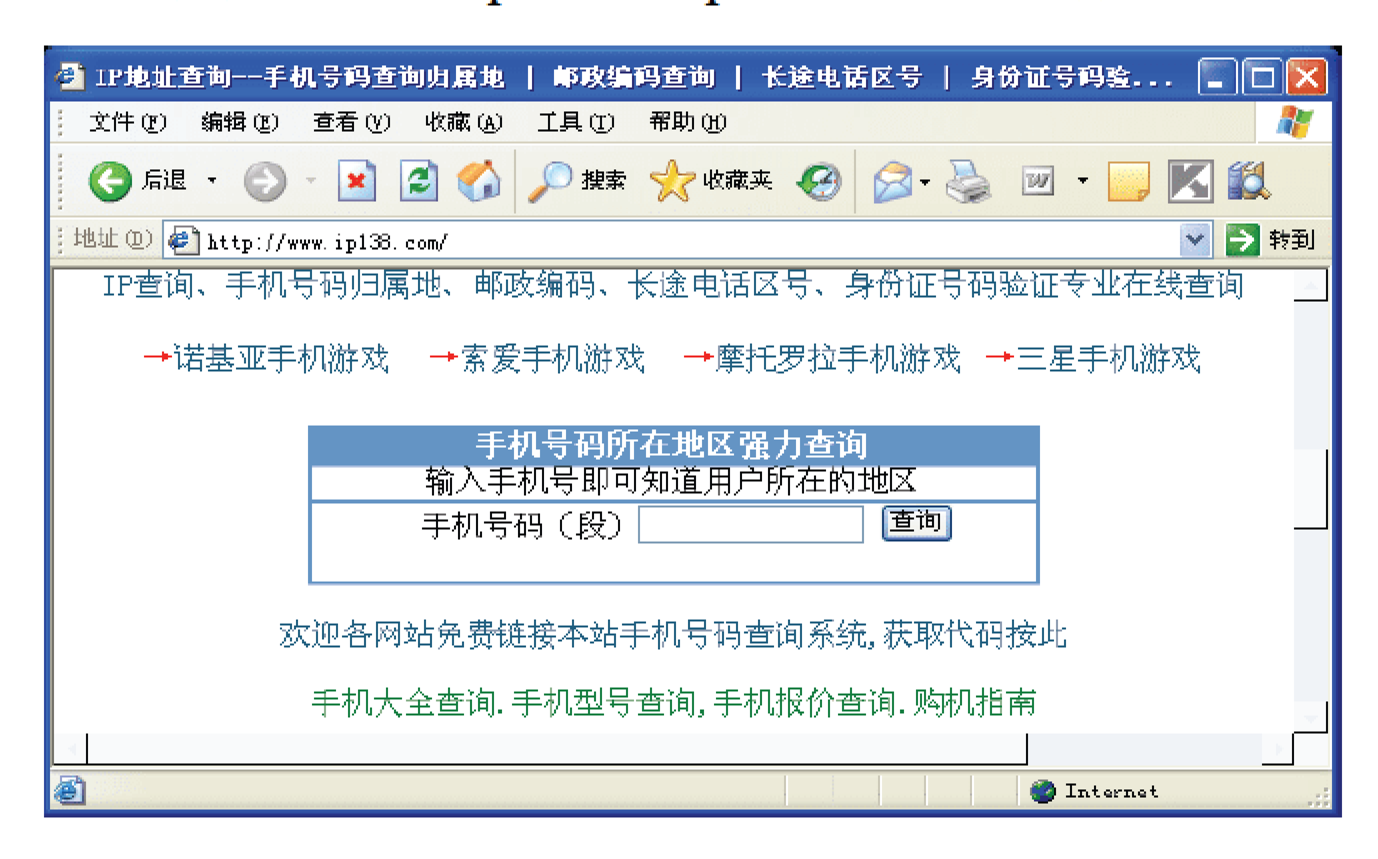


图 25-14 打开网页

（2）在 IE 浏览器中单击【查看】|【源文件】命令，打开【记事本】窗口。在该窗口中显示了图 25-14 所示网页的 HTML 代码。

（3）在【记事本】窗口中单击主菜单【编辑】|【查找】命令，查找字符串“手机号码（段）”（图 25-14 所示查询手机文字框前面的提示文字）。可在 HTML 代码中找到以下一段代码：

```
<form action="http://www.ip138.com:8080/search.asp" method="post"
onsubmit="return checkMobile();" target="mobilewindow" name="mobileform">
<input type="hidden" name="action" value="mobile">
<div align="center"><center><p>  手机号码（段）
<input class="TextBoxStyle" maxLength="11" name="mobile" size="15">
<input class="ButtonStyle" type="submit" value="查询"></p>
</center></div>
</form>
```

其中标签<form>和</form>组成一个表单。在<form>标签中定义了处理提交参数的网页为“http://www.ip138.com:8080/search.asp”，参数提交方式为 POST。需要提交两个参数，一个参数名为 action，其值为 mobile，该行 HTML 代码如下：

```
<input type="hidden" name="action" value="mobile">
```

另一个参数为 mobile，表示用户要查询的手机号码，该行 HTML 代码如下：

```
<input class="TextBoxStyle" maxLength="11" name="mobile" size="15">
```

（4）知道处理数据的网页地址和需要提交的参数后，即可编写如下 VBA 代码：

```
Sub 查询手机所在地()
    Dim str As String, strURL As String

    str = Application.InputBox(prompt:="请输入手机号码：", _
        Title:="手机所在地查询", Type:=2)
    If str = "False" Then Exit Sub
```

```
    If Left(str, 2) <> "13" Or Len(str) <> 11 Then
        MsgBox "请输入正确的手机号码！", vbCritical + vbOKOnly, "提示"
        Exit Sub
    End If
    With Worksheets.Add
        .Name = str
    End With
    strURL = "URL;http://www.ip138.com:8080/search.asp"
    With ActiveSheet.QueryTables.Add(Connection:=strURL, Destination:=
    Range("A2"))
        .Name = "mobile"
        .PostText = "action=mobile&mobile=" & str
        .FieldNames = True
        .RowNumbers = False
        .BackgroundQuery = True
        .Refresh
    End With
End Sub
```

提示：要使用 POST 方式提交多个参数，可将各参数与符号（&）连接起来，再赋值给 PostText 属性即可。

运行以上代码，将弹出如图 25-15 所示对话框，输入手机号码，单击【确定】按钮，即可查得查询的结果，如图 25-16 所示。



图 25-15　输入手机号码

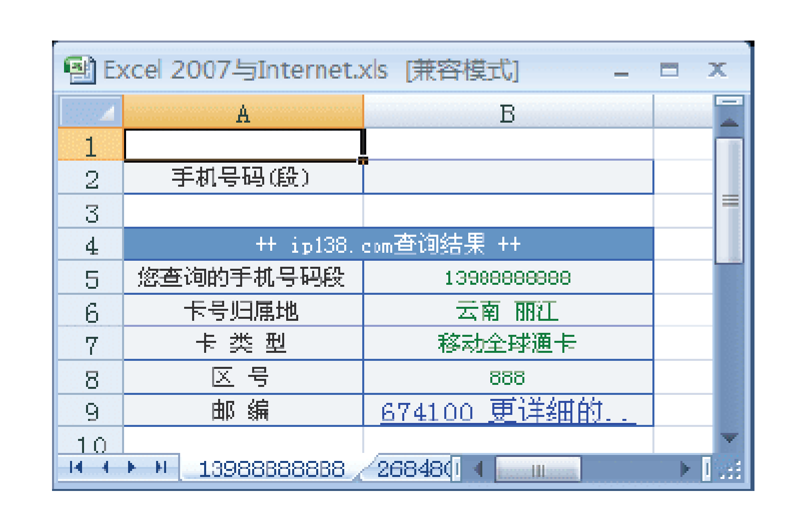


图 25-16　查询结果

25.4　发布数据到 Internet

Excel 的工作簿可以很方便地保存为 HTML 文件。将工作簿保存为 HTML 格式后，可以在网页浏览器中查看电子表格。Excel 还能够将数据和图表保存为交互式网页。

25.4.1　保存为网页

将工作簿保存为 HTML 文件最简单的方式是直接将其保存为网页格式，具体步骤如下：

（1）单击【Office 按钮】打开下拉菜单，单击【另存为】命令打开如图 25-17 所示的【另存为】对话框。

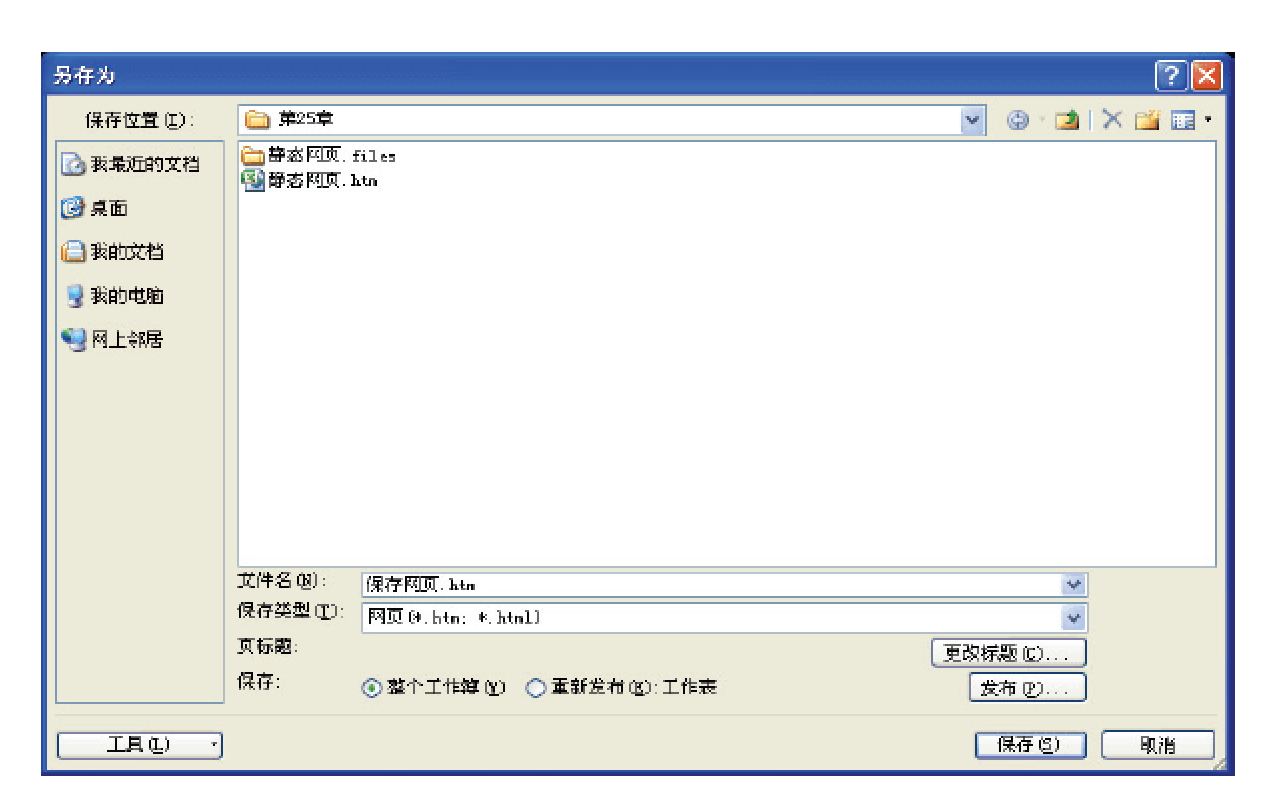


图 25-17 【另存为】对话框

（2）在【保存类型】下拉列表框中选择“网页”，单击【更改标题】按钮可修改网页的标题。单击【保存】按钮即可将工作簿保存为网页格式。

（3）用 IE 浏览器打开保存的网页，如图 25-18 所示。

在图 25-18 所示网页中，显示了工作表中的数据和图表。下方显示了 3 个工作表的标签，单击标签名可在各工作表之间进行切换。

（4）在图 25-17 所示的【另存为】对话框中，单击【发布】按钮将打开如图 25-19 所示的【发布为网页】对话框。在该对话框中可选择要发布的工作表、单元格区域，单击【发布】按钮，可将选择的目标发布为网页文件。

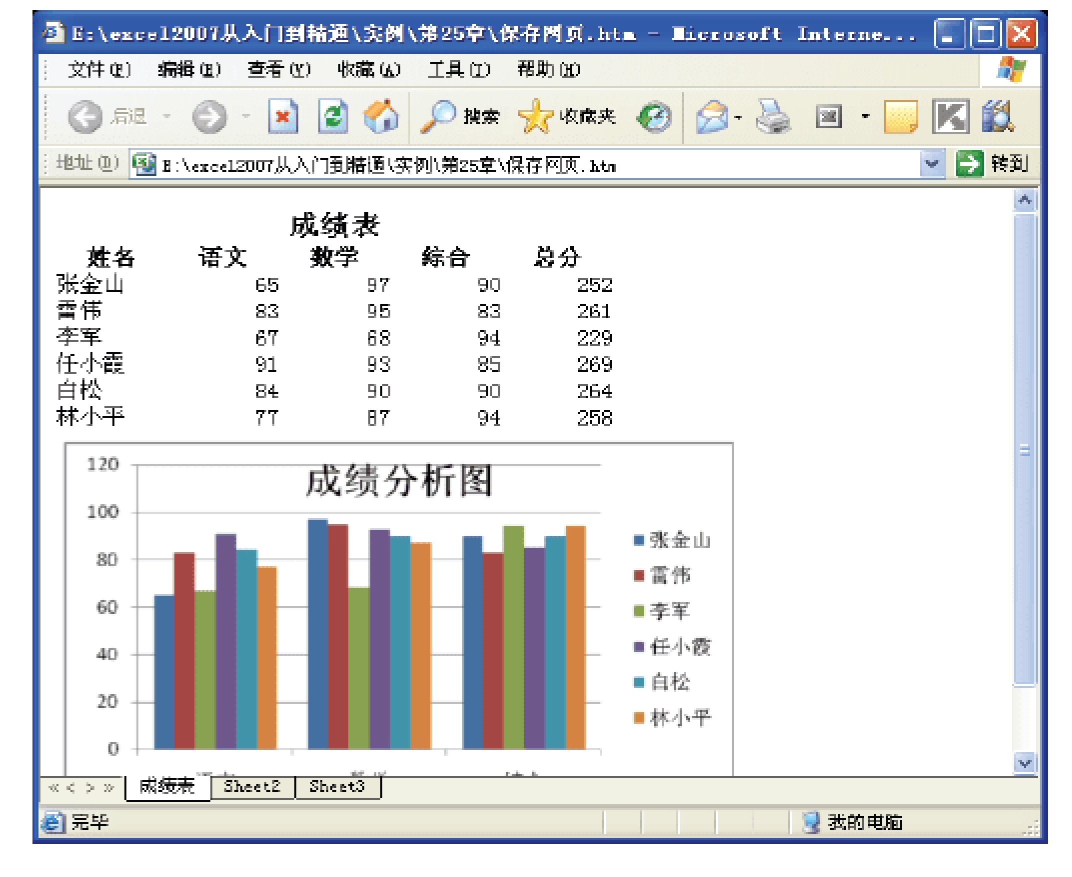


图 25-18 打开网页

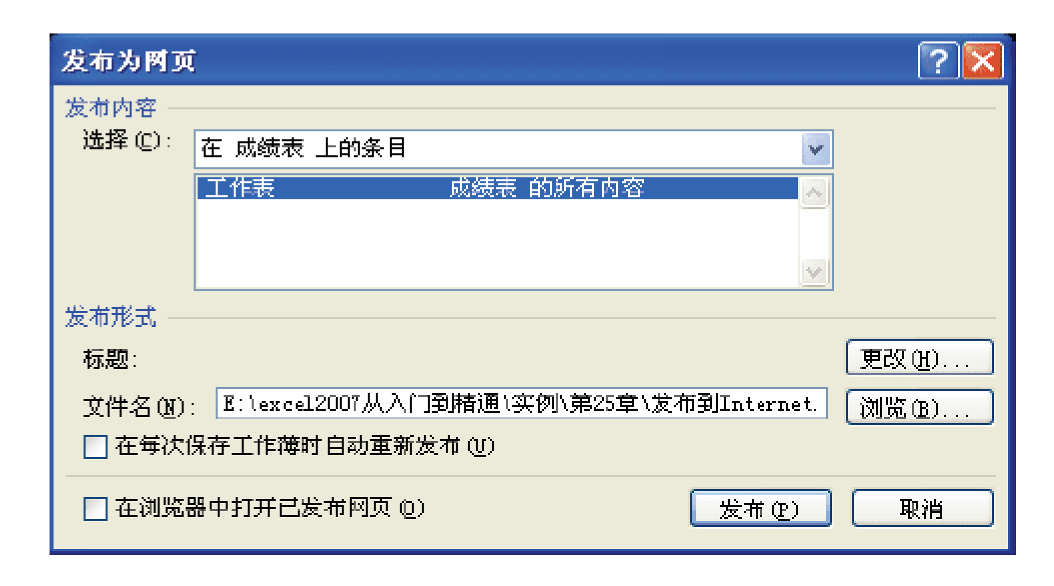


图 25-19 【发布为网页】对话框

25.4.2 用 VBA 代码发布网页

在 VBA 对象模型中，PublishObject 对象代表保存为网页的工作簿中的元素，并可以

根据 PublishObject 对象中属性和方法所指定的值进行刷新。PublishObject 对象是 PublishObjects 集合对象的成员。

使用 PublishObjects 集合的 Add 方法，可创建一个对象，该对象代表保存在网页上的文档中的项目。其语法格式如下：

```
表达式.Add(SourceType, FileName, Sheet, Source, HtmlType, DivID, Title)
```

各参数的含义如下所述。

- SourceType：设置发布对象的类型，可设置为以下常量。
  - xlSourceWorkbook：工作簿；
  - xlSourceSheet：整张工作表；
  - xlSourcePrintArea：选定的用于打印的单元格区域；
  - xlSourceAutoFilter："自动筛选"区域；
  - xlSourceRange：单元格区域；
  - xlSourceChart：图表；
  - xlSourcePivotTable：数据透视表；
  - xlSourceQuery：查询表（外部数据区域）。
- FileName：用于保存发布对象的 URL 或路径（本地或网络）。
- Sheet：保存为网页的工作表名称。
- Source：用来标识数据项的唯一名称，该参数取决于 SourceType 参数的设置。如果 SourceType 是 xlSourceRange，Source 为单元格区域或名称；如果 SourceType 为 xlSourceChart、xlSourcePivotTable 或 xlSourQuery，则 Source 为图表名称、数据透视表名或查询名称。
- HtmlType：指定项目是保存为交互式的 Microsoft Office Web 组件还是保存为静态文本和图像。可以为以下常量：
  - xlHtmlStatic：使用静态（非交互式）HTML，仅用于查看，不允许修改数据。
  - xlHtmlCalc：使用电子表格组件，该组件允许在 IE 上查看、操作表格中的数据。
  - xlHtmlList：使用数据透视表组件，允许在 IE 上重新排列、筛选和汇总信息。
  - xlHtmlChart：使用图表组件，允许在 IE 中创建交互式图表。
- DivID：在 HTML DIV 标记中使用的唯一标识符，它用于标识网页上的项。
- Title：网页的标题。

使用 Add 方法创建 PublishObject 对象后，还需要使用 Publish 方法将文档中的项或项的集合保存到网页中。Publish 方法的语法格式如下：

```
表达式.Publish(Create)
```

参数 Create 为 True 时，若设置的 HTML 文件已经存在，将覆盖该文件。

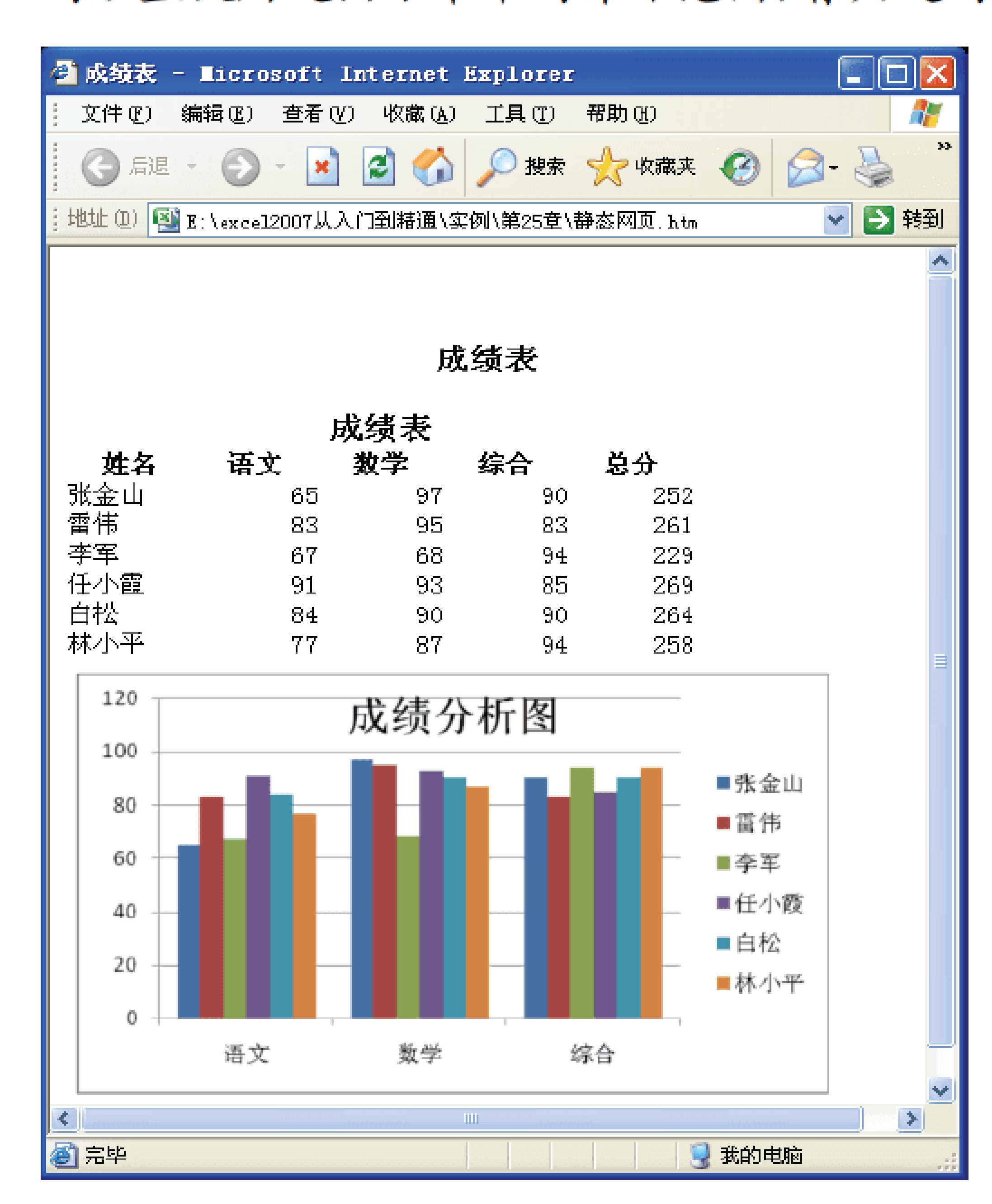
了解 PublishObject 对象的方法后，即可编写程序发布网页。例如，以下代码将工作表"成绩表"中的图据和图表发布为一个静态网页：

```
Sub 发布为静态网页()
    Dim pub As PublishObject
```

```
    Set pub = ThisWorkbook.PublishObjects.Add( _
        SourceType:=xlSourceSheet, _
        Filename:=ThisWorkbook.Path & "\静态网页.htm", _
        Sheet:="成绩表", _
        HtmlType:=xlHtmlStatic, _
        Title:="成绩表")
    pub.Publish True              '发布网页
End Sub
```

执行以上程序后，将在工作簿所在文件夹中创建一个名为“静态网页.htm”的 HTML 文件，同时还自动创建一个名为“静态网页.files”的子文件夹。用 IE 浏览器打开该网页，可看到如图 25-20 所示的效果，在该网页中显示了工作表“成绩表”中的数据和图表，用户不能修改网页中的数据。

提示： 在 Excel 2000 至 Excel 2003 中，可以使用 PublishObject 创建交互式网页。在这种交互式网页中使用 Office Web 组件，让用户可以在 IE 中操作 Excel 工作表中的数据。在 Excel 2007 中，Office Web 组件已经被删除，取而代之的是一个专用服务器端 Excel 组件。该组件运行于 SharePoint 服务器中，并用网页的形式显示给用户。通过该组件可以进行打开 Excel 工作簿、更新数据、执行计算等操作。有关 SharePoint 的配置使用超出了本书的介绍范围，有兴趣的读者可参考相关书籍。



姓名	语文	数学	综合	总分
张金山	65	97	90	252
雷伟	83	95	83	261
李军	67	68	94	229
任小霞	91	93	85	269
白松	84	90	90	264
林小平	77	87	94	258

图 25-20　发布的网页

第 6 部分　VBA 高级应用

VBA 作为一种简单的宏设计语言，相对来说比较简单。但可以通过其他程序设计语言非常方便地扩充其功能。例如，可用 VBA 制作 Excel 加载宏、使用 VB 等高级程序语言设计加载项、通过类模块自定义对象、调用 Windows API 函数等。

本部分共 5 章，分别介绍了在 VBA 中，使用这些高级应用的方法。

第 26 章　使用 Excel 加载宏

在 Excel 中，通过加载宏可以扩展 Excel 的功能。加载宏是为 Excel 提供自定义命令或自定义功能的补充程序。本章将介绍创建和使用加载宏的方法。

26.1　加载宏的概念

加载宏是一类程序，通过加载宏可为 Microsoft Excel 添加可选的命令和功能，以扩充 Excel 的功能。

26.1.1　加载宏的类型

根据创建加载宏的方式不同，可以将加载宏分为多种类型，常用的有以下两种：

1．Excel加载宏

这是最常见的加载宏，由 Excel VBA 创建。在 Excel 2007 之前版本中，加载宏的扩展名为*.xla，Excel 2007 加载宏的扩展名为*.xlam。大部分以前版本的加载宏也可以在 Excel 2007 中加载使用。

在 Excel 工作簿编写好 VBA 代码后，将其保存为扩展名.xla 或.xlam 后，该工作簿就成为一个 Excel 加载宏。Excel 加载宏与 Excel 工作簿主要有以下两点区别：

- 在 VBE 中查看 ThisWorkbook 对象的属性，Excel 工作簿的 IsAddin 属性为 False，如图 26-1 所示。Excel 加载宏的 IsAddin 属性为 True。如图 26-2 所示。

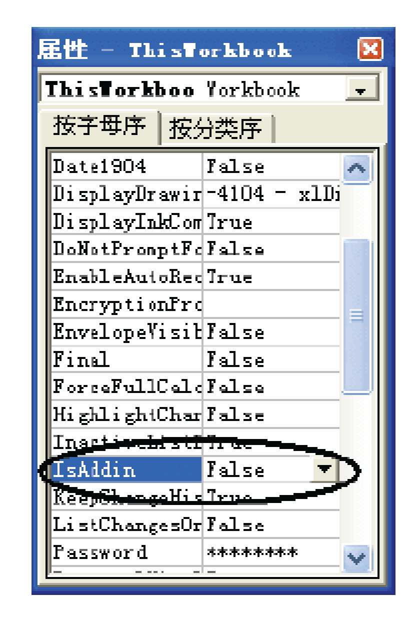




图 26-1　Excel 工作簿属性

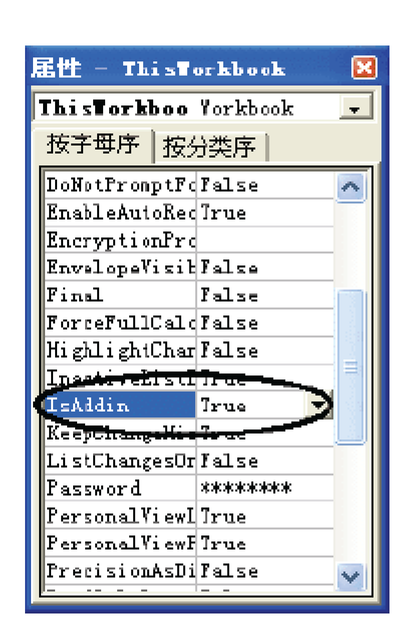


图 26-2　Excel 加载宏属性

- Excel 加载宏处于隐藏状态，并且不能通过【视图】选项卡【窗口】组中的【取消

隐藏】按钮将其显示出来。

2．自定义的组件对象模型（COM）加载宏。

COM 加载宏使用各种编程语言（例如，VB、VC 等）编写，并编译生成扩展名为.dll 的文件。开发这类加载宏需要开发者掌握相应的编程语言，并按照一定的方式编写加载宏的事件过程代码。

26.1.2　加载宏的用途

加载宏的主要作用有：

- ❑ 保存自定义函数，载入加载项后，用户可以像使用内置函数一样调用这些函数。
- ❑ 保存专用宏，如果不希望用户查看或修改宏，可将这些宏保存在加载项文件中，并使用密码保护加载项文件。
- ❑ 扩展 Excel 功能，使用 VBA 创建一些分析工具，用于扩展 Excel 的功能。

与普通工作簿或个人工作簿中的宏相比，加载宏有诸多优越之处。首先，加载宏能方便地提供给他人使用（简单的复制文件即可），让宏的开发者与使用者可以完全分离。因此，开发人员可以为某一目的编写包含大量复杂处理、功能完备的加载宏，使用者只需要像使用 Excel 的标准命令一样进行操作；其次，加载宏的按需加载、卸载机制，有利于系统内存的有效利用。

26.1.3　Excel 中已有的加载宏

在互联网上可以下载很多加载宏程序供用户下载使用。另外，在安装 Excel 时，已向系统中安装了很多加载宏程序，下面简单介绍这些加载宏的功能。

- ❑ Internet Assistant VBA：开发人员可用 Internet Assistant 语法，将 Excel 数据发布到网站上。
- ❑ 查阅向导：帮助创建在列表中查找数据的公式。
- ❑ 分析工具库：提供一组包括金融、统计和工程类的数据分析工具和函数，增添了 Excel 中没有包含的统计和分析功能。主要有方差分析、相关系数、协方差、描述统计、指数平滑、F-检验 双样本方差、傅利叶分析、直方图、移动平均、随机数发生器、排位与百分比排位、回归、抽样、t 检验、z 检验等。
- ❑ 分析工具库 — VBA：内容与上面的相同，为分析工具库提供的 VBA 函数。允许开发人员用分析工具库的语法发布金融、统计及工程分析工具和函数。
- ❑ 规划求解加载项：提供了公式求解和优化的工具。对基于可变单元格和条件单元格的假设分析方案进行求解计算。
- ❑ 欧元工具：提供用于欧元转换及格式设置的工具。将数值的格式设置为欧元格式，并提供 EUROCONVERT 工作表函数用于转换货币。
- ❑ 条件求和向导：提供了对列表中的数据根据不同的条件求和的工具。

这些加载宏都是应用到某些特殊领域，普通用户一般用不到。

26.2 管理加载宏

要在工作簿中使用加载宏提供的功能，必须将该加载宏载入内存。本节简单介绍载入加载宏和卸载加载宏的方法。

26.2.1 载入加载宏

在使用加载宏之前，必须先将其载入 Excel 中。其实，载入加载宏就是打开加载宏文件，将相关命令添加到相关选项卡中（对于用户创建的加载宏，默认情况下都是添加到【加载项】选项卡中）。下面介绍载入加载宏的方法。

（1）启动 Excel 后，单击【Office 按钮】打开下拉菜单，单击右下方的【Excel 选项】按钮打开【Excel 选项】对话框。

（2）单击对话框左侧的【加载项】按钮，如图 26-3 所示。在该对话框中可看到活动加载项和非活动加载项。

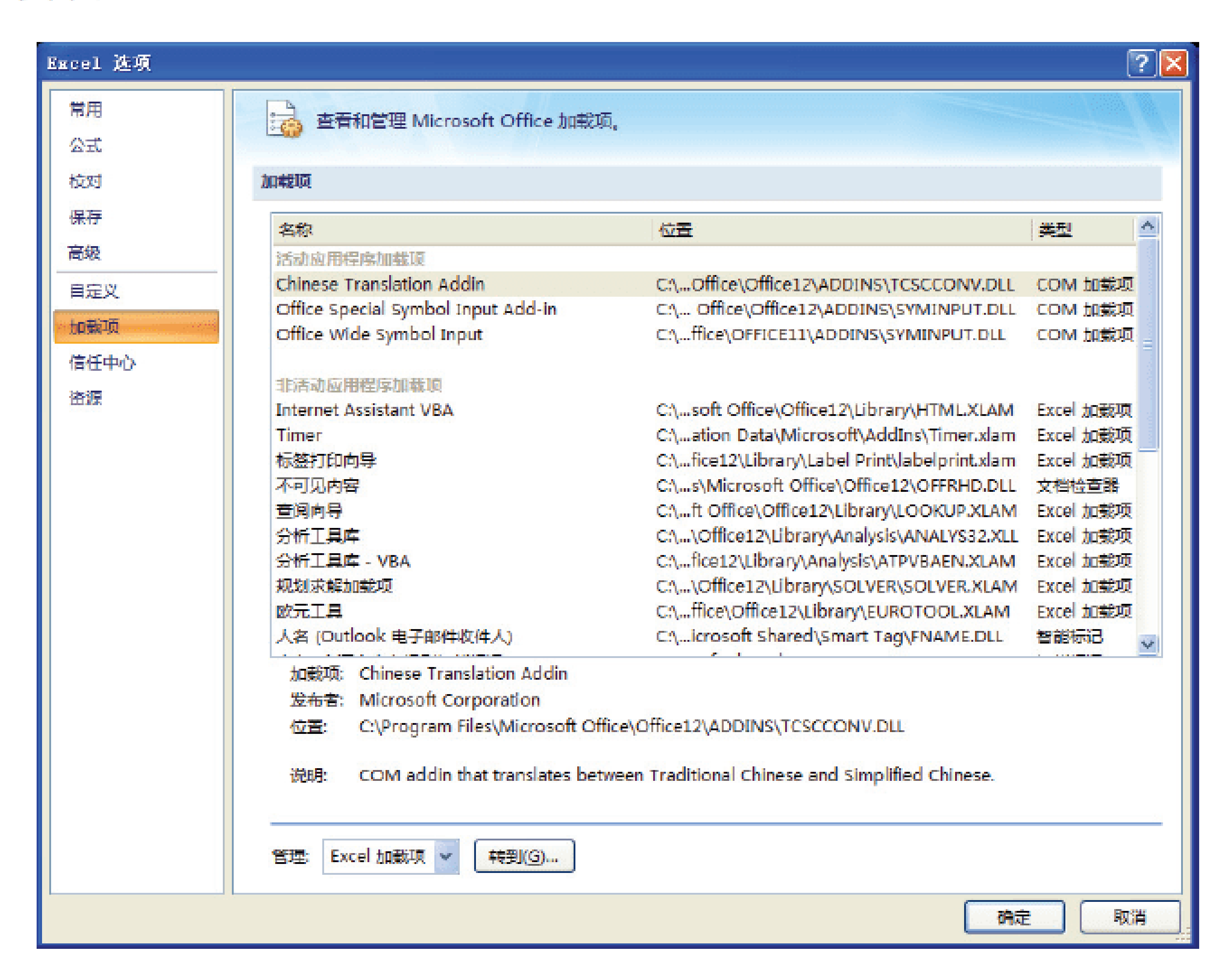


图 26-3 【加载项】选项卡

（3）Excel 2007 支持多种类型的加载项，单击该对话框下方【管理】标签右侧的下拉列表框，在该列表框中选择 Excel 加载项，单击右侧的【转到】按钮，打开如图 26-4 所示的对话框。

（4）在图 26-4 所示对话框中选中【规划求解加载项】，单击【确定】按钮。在【数据】选项卡右侧将增加一个【分析】组，在该组中添加了一个【规划求解】按钮，如图 26-5 所示。

（5）单击功能区上的【规划求解】按钮，即可与使用 Excel 其他内部命令一样的方法

调用【规划求解】命令的功能。

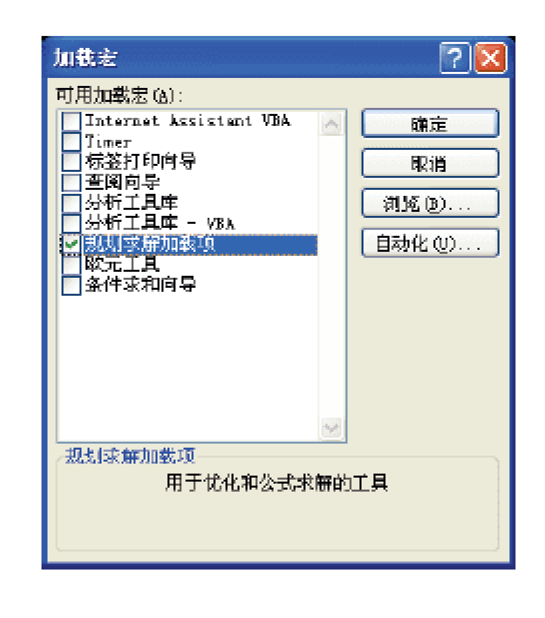


图 26-4　加载宏



图 26-5　功能区

26.2.2　卸载加载宏

将加载宏载入 Excel 后，每次启动 Excel 时，选择的加载宏都会自动被载入。加载宏在运行时将占用一定数量的内存，而且在启动 Excel 时，加载过多的加载宏也会占用太多的启动时间。所以，对于不需要用的加载宏程序，应该及时从 Excel 中卸载掉。

卸载加载宏的方法非常简单，按上面的步骤在图 26-4 所示对话框中去掉相关加载宏前面的勾即可。

加载宏程序一旦被卸载，除非再次加载，否则 Excel 将不会自动进行加载。

26.2.3　系统加载宏列表

在 VBA 对象模型中，使用 AddIn 对象代表单个加载宏。在程序中可通过以下属性获取指定加载宏的属性值。

- ❑ CLSID 属性：获取加载宏的唯一标识符，或识别对象的 CLSID。
- ❑ FullName 属性：返回对象的名称（以字符串表示），包括其磁盘路径。
- ❑ Installed 属性：将此属性设为 True，将安装指定加载宏，并调用 Auto_Add 函数。将此属性设为 False，则移去指定加载宏，并调用 Auto_Remove 函数。
- ❑ Name 属性：返回对象的名称。
- ❑ Path 属性：返回应用程序的完整路径，不包括末尾的分隔符和应用程序名称。

Excel 中所有的 AddIn 对象组成 AddIns 集合，无论加载宏是否装载到内存中，都将包含在 AddIns 集合中。该集合中的对象与图 26-4 所示对话框中显示的加载宏列表对应。

使用以下代码可显示 Excel 中的加载宏列表：

```
Sub 加载宏列表()
    Dim adi As AddIn, i As Long
    i = 1
    With Worksheets("加载宏列表")
    For Each adi In Application.AddIns
        i = i + 1
```

```
        .Cells(i, 1) = adi.Name            '加载宏名称
        .Cells(i, 2) = adi.Installed       '安装状态
        .Cells(i, 3) = adi.Path            '文件位置
    Next
    .Columns.AutoFit
    End With
End Sub
```

以上代码从 Addins 集合中循环显示每个加载宏的名称和文件位置。执行以上代码，将在工作表中显示加载宏列表，如图 26-6 所示。

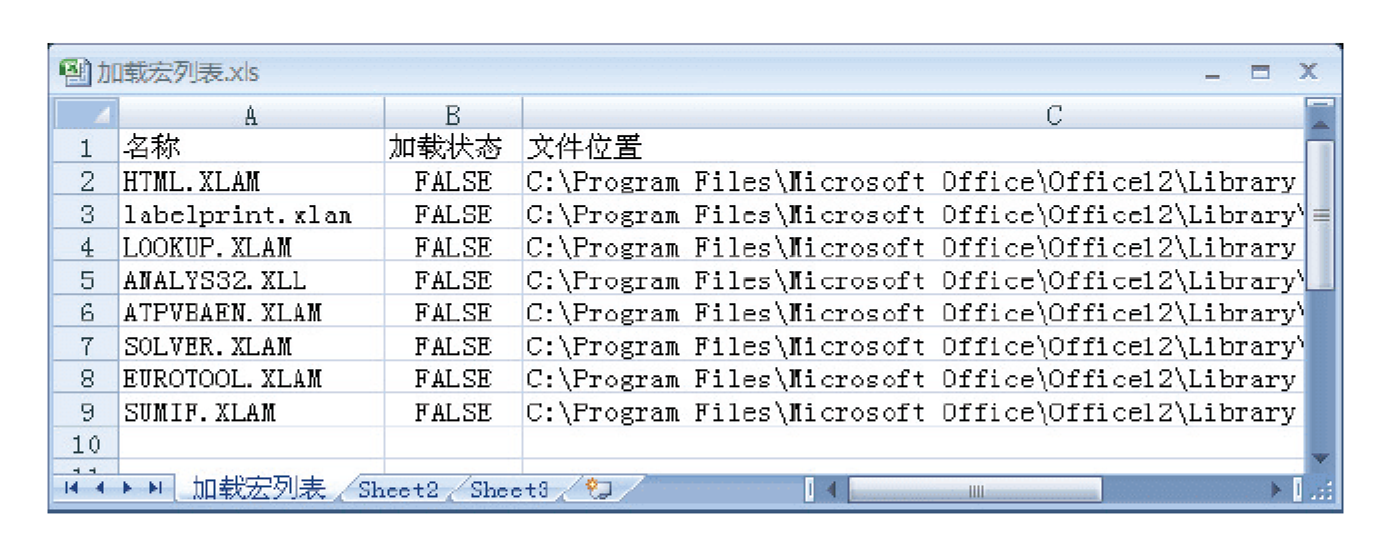


图 26-6　加载宏列表

26.3　创建加载宏

常用的加载宏有两种，即 Excel 加载宏和 COM 加载宏。Excel 加载宏直接在 Excel 中创建，而 COM 加载宏需要使用编程工具（如 VB、C++等）来编写。本节将介绍这两种创建加载宏的方法。

26.3.1　创建 Excel 加载宏

本节以实例形式介绍创建 Excel 加载宏的方法。下面的实例将在加载宏中创建一个“计算个人所得税”的函数，用来扩充 Excel 的功能。

不需用其他软件就可直接从 Excel 工作簿创建加载宏。任何 Excel 工作簿文件都可以转换为加载宏，但并不是所有的工作簿文件都适合用于加载宏，一般将包含通用功能的工作簿转换成加载宏。创建 Excel 加载宏的步骤如下：

（1）新建一个 Excel 文档。

（2）按快捷键 Alt+F11 进入 VBE，向工程中插入一个模块。

（3）在模块中编写实现所需功能的子过程。在工作簿的加载宏事件中编写代码。

（4）设置工作簿的属性，其中标题名即为加载宏的名称，备注栏中的说明即为对加载宏功能的描述，当选中这个加载宏时，这些说明将出现在加载宏对话框的底部。

（5）使用另存为命令保存工作簿。保存类型选择为加载宏。在 Excel 2007 中可将工作簿保存为扩展名为 Excel 加载宏（扩展名为*.xlam），或 Excel 97-200 加载宏（扩展名为*.xla）

在另存工作簿时，当选择保存为加载宏后，其保存位置将自动切换到用户数据文件夹的 AddIns 文件夹中，如图 26-7 所示。

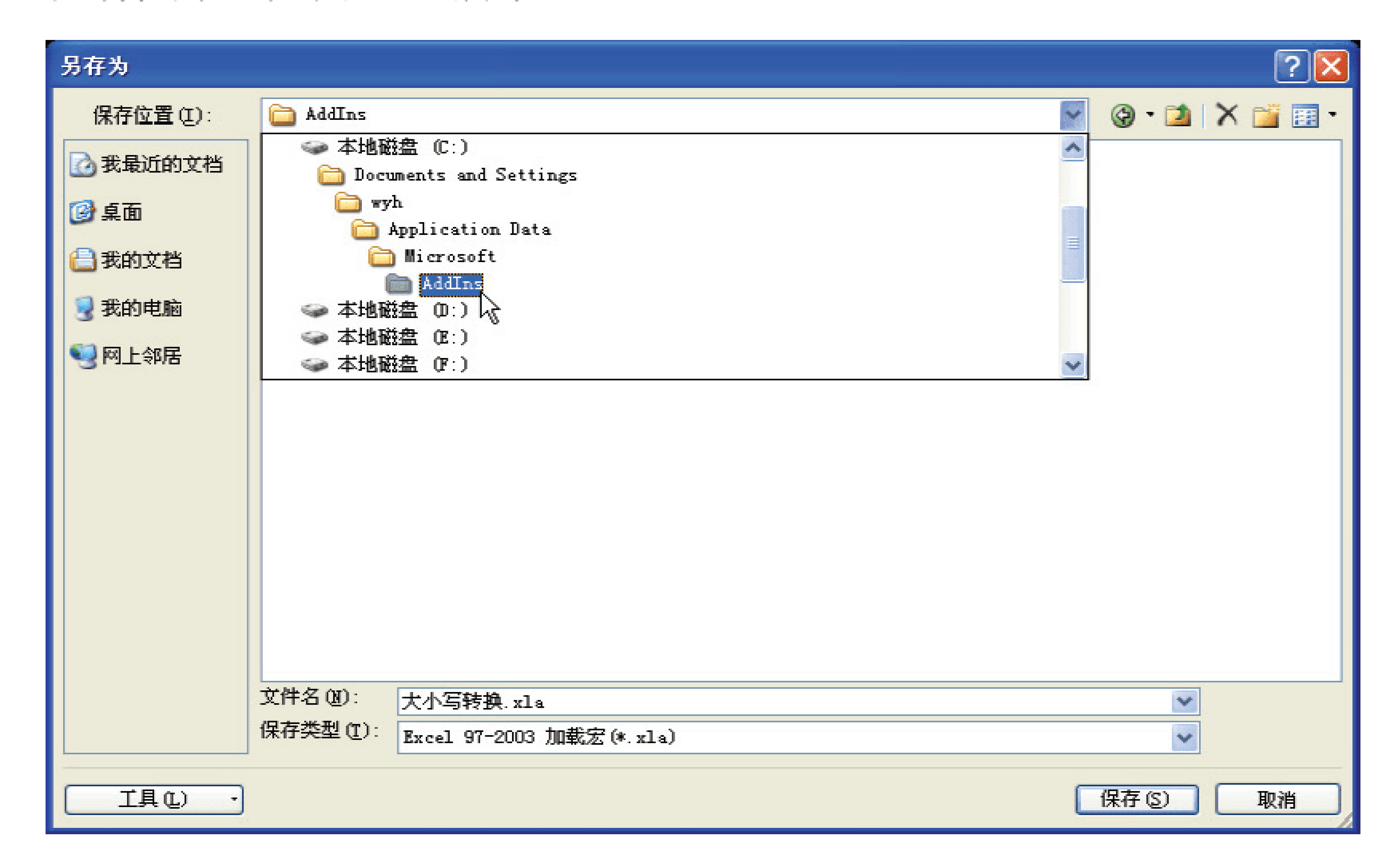


图 26-7　加载宏保存位置

注意：转换成加载宏的工作簿必须至少含有一个工作表，且工作表须处于活动状态。

创建“个人所得税”加载宏的具体方法如下：

（1）新建一个 Excel 工作簿。

（2）按快捷键 Alt+F11 进入 VBE。

（3）在 VBE 中插入一个模块，在模块中输入以下代码，定义 Tax 函数用来计算个人所得税：

```
Option Explicit
Function Tax(Income As Double, Optional Tax_Standard As Integer = 2000) As
Single
    Dim SngBase As Single    '应纳税金额
    Dim SngRate As Single    '税率
    Dim intTemp As Integer   '速算扣除数

    SngBase = Income - Tax_Standard
    Select Case SngBase
        Case Is <= 0
            SngRate = 0
            intTemp = 0
        Case Is <= 500
            SngRate = 0.05
            intTemp = 0
        Case Is <= 2000
            SngRate = 0.1
            intTemp = 25
```

```
        Case Is <= 5000
            SngRate = 0.15
            intTemp = 125
        Case Is <= 20000
            SngRate = 0.2
            intTemp = 375
        Case Is <= 40000
            SngRate = 0.25
            intTemp = 1375
        Case Is <= 60000
            SngRate = 0.3
            intTemp = 3375
        Case Is <= 80000
            SngRate = 0.35
            intTemp = 6375
        Case Is <= 100000
            SngRate = 0.4
            intTemp = 10375
        Case Is > 100000
            SngRate = 0.45
            intTemp = 15375
    End Select

    Tax = Round(Abs(SngBase * SngRate - intTemp), 2)
End Function
```

在以上代码中，函数 Tax 的参数中使用了 Optional 关键字。使用该关键字表示参数不是必需的，在调用该函数时可以不输入该参数。可选参数可设置默认值，具体形式如下：

```
Function Tax(Income As Double, Optional Tax_Standard As Integer = 2000) As
Single
```

（4）返回 Excel 界面，在工作表中选择一个单元格，输入以下公式测试函数的运行状态：

```
=tax(4000)              '以默认值 2000 为个税起征点
=tax(4000,1600)         '以 1600 为个税起征点
```

经过多次测试，确认自定义的函数运行正确，即可进行以下操作。

（5）单击【Office 按钮】打开下拉菜单，选择菜单【准备】|【属性】命令，在工作簿上方打开【文档属性】面板，修改工作簿的属性如图 26-8 所示。



图 26-8　修改属性

（6）单击【Office 按钮】打开下拉菜单，选择菜单【另存为】命令，将工作簿另存为

加载宏“个人所得税.xla”，完成加载宏的创建。

下节将介绍调用加载宏所提供函数的方法。

26.3.2　创建 COM 加载宏

Excel 加载宏是扩展名为*.xla 或*.xlam 的 Excel 工作簿，可在 Excel 中打开进行修改。而 COM 加载宏是一个二进制文件，其扩展名为*.DLL。创建 COM 加载宏需要使用程序设计语言（如 VB、C++等）。

用 VB 创建 COM 加载宏时，应在 VB 中引用以下类型库：

- ❑ Visual Basic for Applications；
- ❑ OLE Automation；
- ❑ Microsoft Add-in Designer；
- ❑ Microsoft Office 12.0 Object Library；
- ❑ Microsoft Excel 12.0 Object Library。

提示：若使用 Office 2003，则引用 Office 11.0 和 Excel 11.0 类型库。

在 VB 中添加以上类型库的引用后，就可以在 VB 程序中引用 Excel 的对象模型，从而控制 Excel 程序。在 VB 程序中使用以下语句创建与 Excel 的 Application 的连接：

```
Set oXL = Excel.Application
```

在 VB 中创建 COM 加载宏的步骤如下：

（1）用 VB 创建一个“外接程序”工程。

（2）在工程的“设计器”Connect 代码中添加以下代码，响应用户单击工具按钮（或菜单）的事件。

```
Dim WithEvents MyButton As Office.CommandBarButton
```

（3）在 AddinInstance_OnConnection 事件过程中编写代码，用于创建工具按钮（或菜单），或者在此处进行一些基础设置（如创建与 Excel 的 Application 对象的连接等）。

（4）在 AddinInstance_OnDisconnection 事件过程中编写代码，清除加载宏的各种设置。

（5）编写按钮对象 MyButton 的单击事件，处理用户单击按钮时执行的子过程。

```
Private Sub MyButton_Click(ByVal Ctrl As Office.CommandBarButton, _
CancelDefault As Boolean)
```

提示：如果没有创建工具按钮或菜单，本步骤可省略。

（6）编写完成相应功能的子过程代码。

（7）将 VB 工程编译为 DLL 文件，完成 COM 加载宏的创建过程。

下面以具体实例演示创建 COM 加载宏的方法。该实例完成的功能很简单：将单元格中的英文字母进行大小写相互转换。

本例使用 VB6 创建 COM 加载宏，要求计算机系统中首先要安装 VB6 企业版。具体

的创建过程如下：

（1）启动 VB6，在【新建工程】对话框中选择【外接程序】项，如图 26-9 所示。单击【确定】按钮创建新的工程，进入 VB 开发环境。



图 26-9 【新建工程】对话框

（2）在 VB 开发环境右侧的【工程】管理器窗口中有一个窗体和一个设计器，如图 26-10 所示。用鼠标右键单击窗体 frmAddIn 项，在弹出的快捷菜单中选择【移除 frmAddIn】命令，将窗体删除，只保留设计器项。

（3）在图 26-10 所示【工程】窗口中单击工程名称 MyAddIn，在【属性】窗口中将工程名称改为 Caps，如图 26-11 所示。

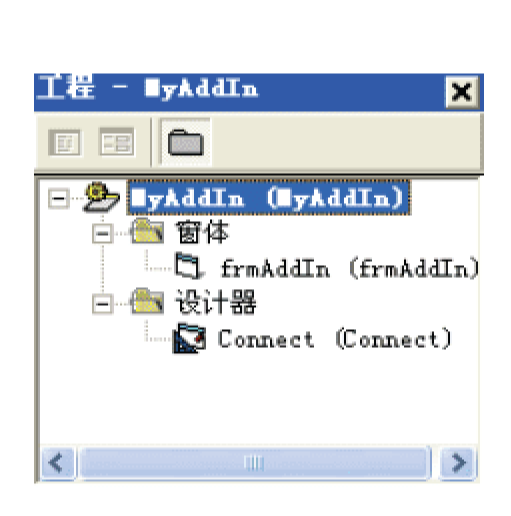


图 26-10 工程管理器



图 26-11 【属性】窗口

（4）双击设计器下的 Connect 选项，打开如图 26-12 所示的对话框。在该对话框中设置 COM 加载项的各项配置参数，如设置加载宏的显示名称、描述、应用到哪类程序及程序的版本、加载宏的启动方式等。

（5）右键单击设计器的 Connect 选项，在弹出的快捷菜单中选择【查看代码】命令，将打开该模块的代码窗口。将代码窗口中已有的代码全部删除。

（6）在代码窗口顶端输入以下代码，定义模块变量。

```
Option Explicit
```

```
Dim oXL As Object                                      '引用 Application 的变量
Dim WithEvents MyButton As Office.CommandBarButton
```



图 26-12　设置加载宏配置参数

以上代码使用 WithEvents 关键字说明 MyButton 是一个用来响应由 ActiveX 对象触发的事件的对象变量。在本例中，用来响应用户单击工具栏按钮的事件。

（7）编写 AddinInstance_OnConnection 事件过程的代码如下：

```
Private Sub AddinInstance_OnConnection(ByVal Application As Object, _
  ByVal ConnectMode As AddInDesignerObjects.ext_ConnectMode, _
  ByVal AddInInst As Object, custom() As Variant)
   On Error Resume Next

   Set oXL = Application

   Set MyButton = oXL.CommandBars("Standard").Controls.Add(1)
   With MyButton
     .Caption = "字母大小写转换"
      .Style = msoButtonCaption
      .Tag = "字母大小写转换"
      .Visible = True
   End With
End Sub
```

AddinInstance_OnConnection 事件过程在加载宏被加载时执行。以上代码首先获取对 Excel 的 Application 对象的引用，接着在 Standard 工具栏中添加一个按钮，并将按钮的引用保存到模块变量 MyButton 中，最后设置按钮的相关属性。

注意：按钮的属性中没有设置 OnAction 属性。

（8）编写 AddinInstance_OnDisconnection 事件过程的代码如下：

```
Private Sub AddinInstance_OnDisconnection(ByVal RemoveMode As _
   AddInDesignerObjects.ext_DisconnectMode, custom() As Variant)
```

```
    On Error Resume Next

    MyButton.Delete
    Set MyButton = Nothing
    Set oXL = Nothing
End Sub
```

AddinInstance_OnDisconnection 事件过程在加载宏被卸载时执行。以上代码使用模块变量 MyButton 快速删除添加的按钮对象。

（9）编写 MyButton_Click 事件过程的代码如下：

```
Private Sub MyButton_Click(ByVal Ctrl As Office.CommandBarButton, _
  CancelDefault As Boolean)
    LCase2UCase
End Sub
```

单击创建的工具按钮时，将执行上面的代码，调用子过程 LCase2UCase 进行大小写字母的转换。

（10）编写子过程 LCase2UCase 的代码如下：

```
Sub LCase2UCase()
    Dim s As String
    With oXL
       s = .ActiveCell.Value

       If Asc(Left(s, 1)) > 90 Then
           .ActiveCell = UCase(s)
       Else
           .ActiveCell = LCase(s)
       End If
    End With
End Sub
```

以上代码通过模块变量 oXL 引用 Excel 的 Application 对象，再通过该对象的属性 ActiveCell 获取 Excel 当前活动单元格的值，并进行大小写的转换。

注意：要引用 Excel 中的对象，必须在对象前面加上对 Excel 的 Application 对象的引用变量，以限定访问的是 Excel 中的对象。

（11）选择菜单【文件】|【保存工程】命令，保存当前的工程。

（12）选择菜单【文件】|【生成 Caps.dll】命令生成 COM 加载宏。

提示：因为在图 26-12 中设置该 COM 加载宏为启动型，所以打开 Excel 应用程序时，该加载宏将自动启动。如果在 Excel 中使用时发现时加载宏代码需要修改，在 VB 中修改后生成 DLL 文件之前需将 Excel 关闭，否则将出现错误。

通过以上步骤，完成创建 COM 加载宏的过程。下节将介绍引用该加载宏的方法。

26.4　使用加载宏

用户可以使用 VBA、VB、VC 等开发自定义加载宏，也可从互联网上下载加载宏。要使用这些加载宏，需要通过【Excel 选项】对话框将其添加到 Excel 系统中，然后就可以在工作簿中使用 Excel 加载宏提供的功能，与使用 Excel 内置功能相同。

下面分别介绍使用 Excel 加载宏和 COM 加载宏的方法。

26.4.1　使用 Excel 加载宏

自定义加载宏的使用方法与系统自带的加载宏相同，不同的是装载加载宏时需要选择文件保存的位置。下面列出使用前面创建的加载项的步骤。

（1）启动 Excel，单击【Office 按钮】打开下拉菜单，选择右下方的【Excel 选项】按钮，打开【Excel 选项】对话框。

（2）单击对话框左侧的【加载项】按钮，在对话框下方【管理】标签右侧的下拉列表框中选择【Excel 加载项】，如图 26-13 所示。

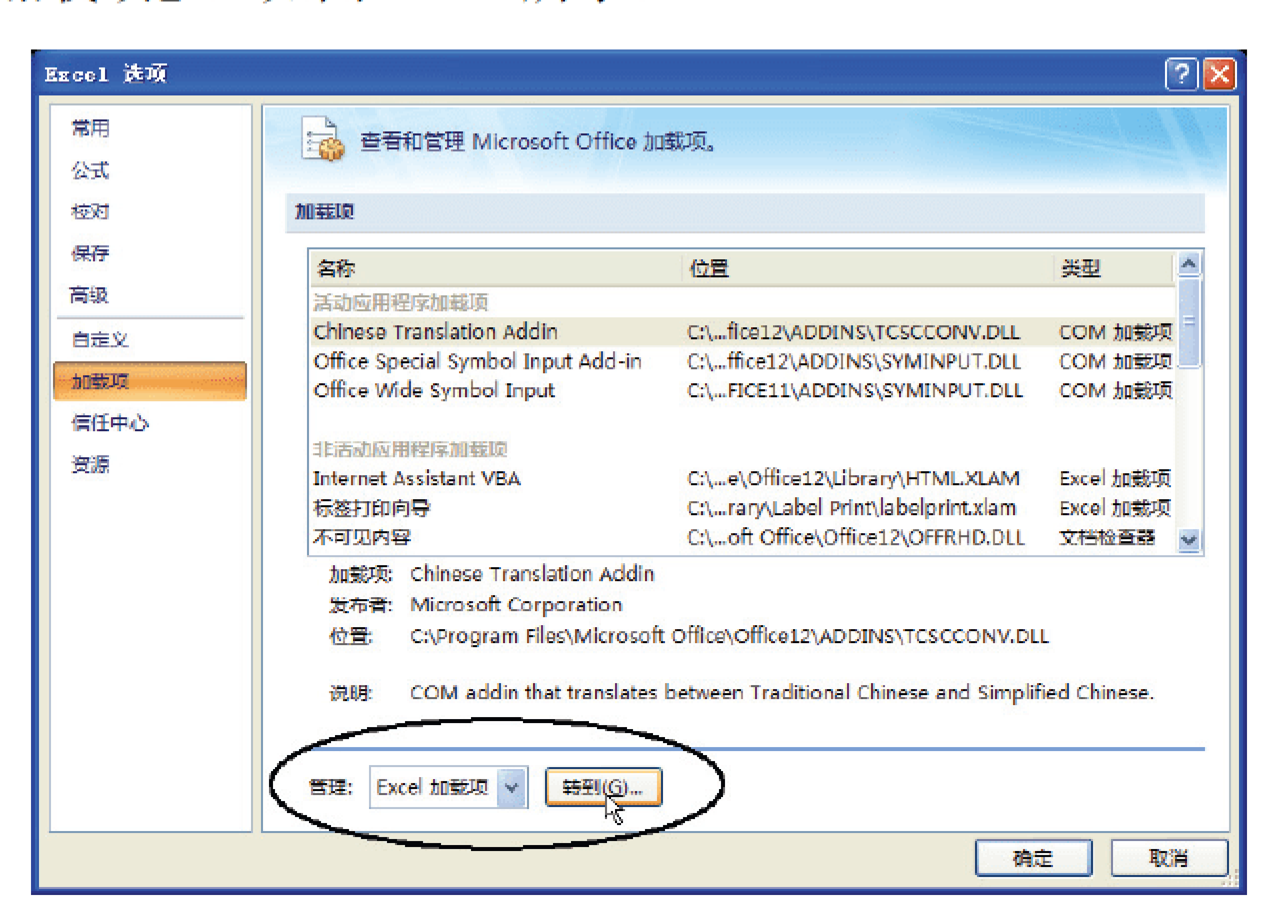


图 26-13　Excel 加载项

（3）在图 26-13 所示对话框中单击右侧的【转到】按钮，打开如图 26-14 所示的对话框，在该对话框中显示了 Excel 能引用的加载宏的列表。

（4）单击对话框右侧的【浏览】按钮，默认情况下将自动定位到用户数据文件夹的 AddIns 文件夹中，如图 26-15 所示。若用户自定义的 Excel 加载宏位于其他位置，可通过该对话框查找到加载宏文件。

（5）如图 26-15 所示，在对话框中选择“个人所得税.xla”加载项文件。

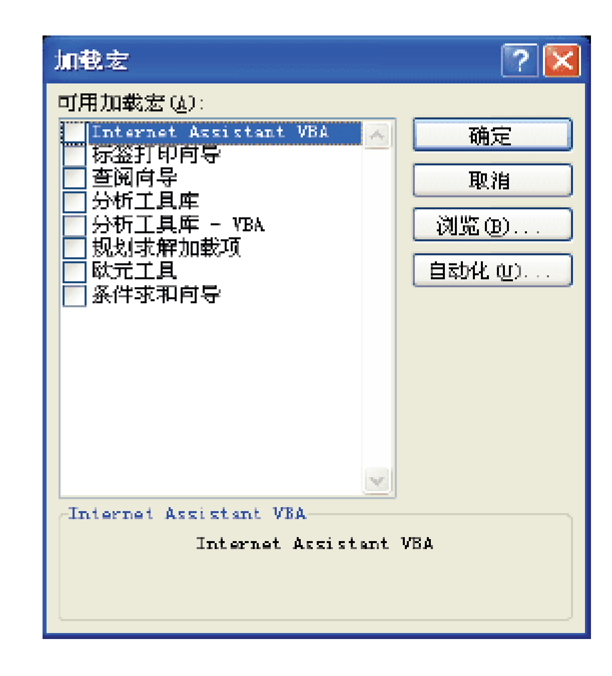


图 26-14　【加载宏】对话框

（6）单击【确定】按钮将该加载宏添加到 Excel 中，在【加载宏】对话框中可以看到【个人所得税】项目已处于选中状态，在下方有该加载宏的提示信息，如图 26-16 所示。

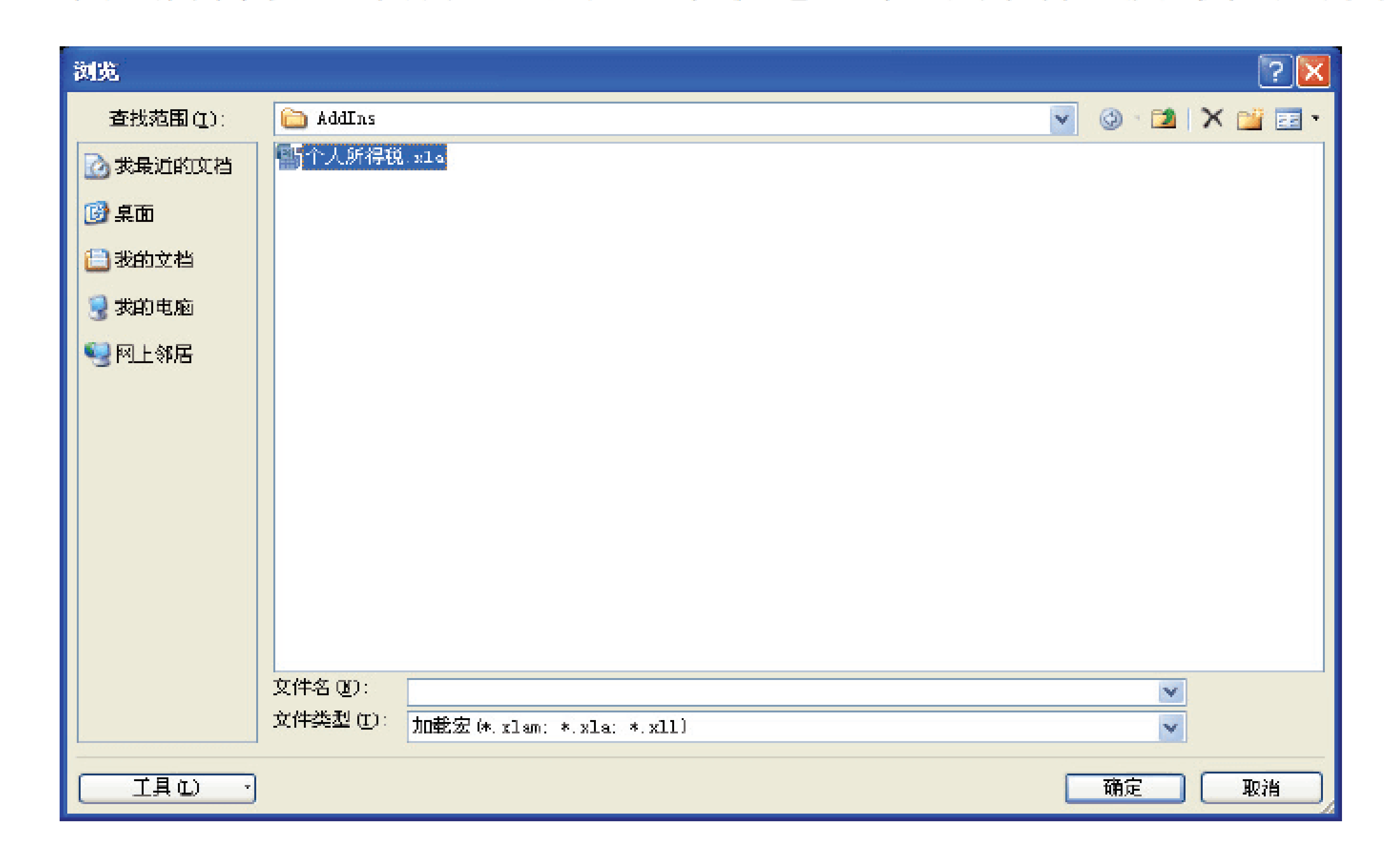


图 26-15　选择加载宏文件

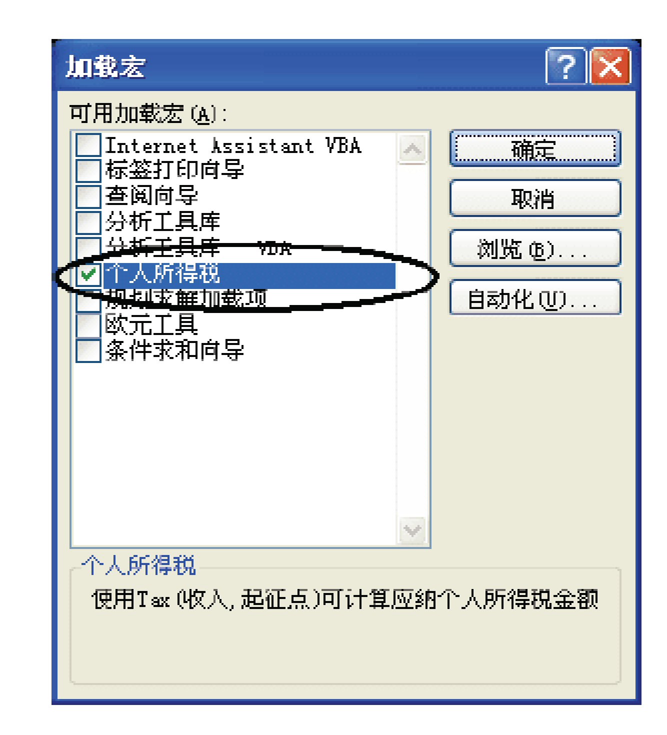


图 26-16　选择加载宏

（7）单击【确定】按钮返回 Excel 中。

（8）打开或创建一个工资表，如图 26-17 所示。在该工作表中，“所得税”列需调用加载宏中的函数 Tax 来进行计算。

工资表.xls [兼容模式]

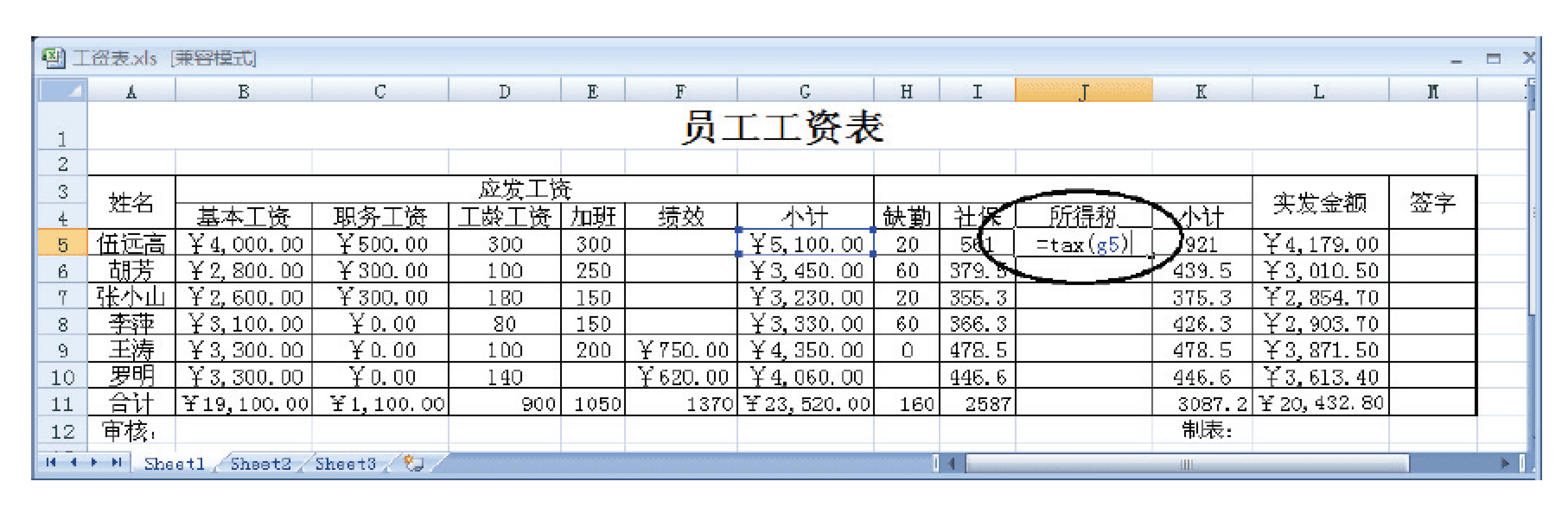
员工工资表

姓名	应发工资										实发金额	签字
	基本工资	职务工资	工龄工资	加班	绩效	小计	缺勤	社保	所得税	小计		
伍远高	￥4,000.00	￥500.00	300	300		￥5,100.00	20	561		[illegible]81	￥4,519.00	
胡芳	￥2,800.00	￥300.00	100	250		￥3,450.00	60	379.5		439.5	￥3,010.50	
张小山	￥2,600.00	￥300.00	180	150		￥3,230.00	20	355.3		375.3	￥2,854.70	
李萍	￥3,100.00	￥0.00	80	150		￥3,330.00	60	366.3		426.3	￥2,903.70	
王涛	￥3,300.00	￥0.00	100	200	￥750.00	￥4,350.00	0	478.5		478.5	￥3,871.50	
罗明	￥3,300.00	￥0.00	140		￥620.00	￥4,060.00		446.6		446.6	￥3,613.40	
合计	￥19,100.00	￥1,100.00	900	1050	1370	￥23,520.00	160	2587		2747.2	￥20,772.80	
审核：										制表：		

Sheet1 Sheet2 Sheet3

图 26-17　工作表

（9）单击选择单元格 J5，在该单元格中输入公式“=tax(g5)”，调用加载宏中定义的函数 Tax 计算个人所得税。如图 26-18 所示。



工资表.xls [兼容模式]

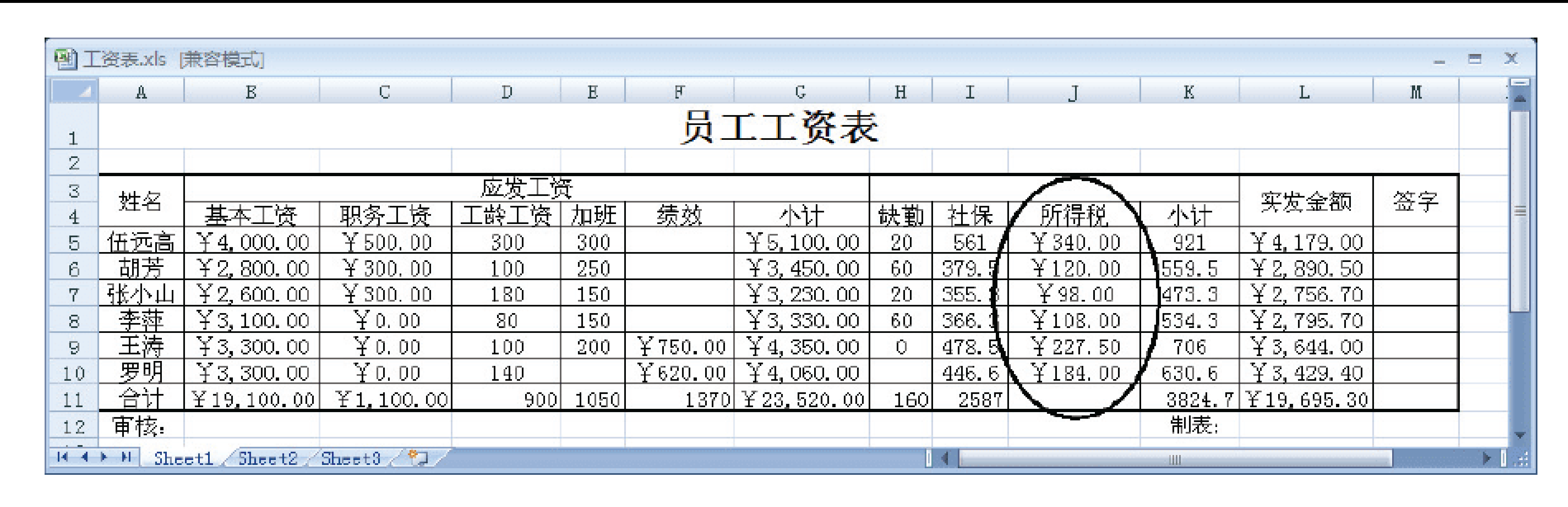
员工工资表

姓名	应发工资										实发金额	签字
	基本工资	职务工资	工龄工资	加班	绩效	小计	缺勤	社保	所得税	小计		
伍远高	￥4,000.00	￥500.00	300	300		￥5,100.00	20	561	=tax(g5)	921	￥4,179.00	
胡芳	￥2,800.00	￥300.00	100	250		￥3,450.00	60	379.5		439.5	￥3,010.50	
张小山	￥2,600.00	￥300.00	180	150		￥3,230.00	20	355.3		375.3	￥2,854.70	
李萍	￥3,100.00	￥0.00	80	150		￥3,330.00	60	366.3		426.3	￥2,903.70	
王涛	￥3,300.00	￥0.00	100	200	￥750.00	￥4,350.00	0	478.5		478.5	￥3,871.50	
罗明	￥3,300.00	￥0.00	140		￥620.00	￥4,060.00		446.6		446.6	￥3,613.40	
合计	￥19,100.00	￥1,100.00	900	1050	1370	￥23,520.00	160	2587		3087.2	￥20,432.80	
审核：										制表：		

Sheet1 Sheet2 Sheet3

图 26-18　输入加载宏中定义的函数

（10）向下拖动单元格 J5 的填充柄，得到 J 列各单元格的公式数据，如图 26-19 所示。



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	员工工资表												
2													
3	姓名	应发工资										实发金额	签字
4		基本工资	职务工资	工龄工资	加班	绩效	小计	缺勤	社保	所得税	小计		
5	伍远高	￥4,000.00	￥500.00	300	300		￥5,100.00	20	561	￥340.00	921	￥4,179.00	
6	胡芳	￥2,800.00	￥300.00	100	250		￥3,450.00	60	379.5	￥120.00	559.5	￥2,890.50	
7	张小山	￥2,600.00	￥300.00	180	150		￥3,230.00	20	355.[illegible]	￥98.00	473.3	￥2,756.70	
8	李萍	￥3,100.00	￥0.00	80	150		￥3,330.00	60	366.[illegible]	￥108.00	534.3	￥2,795.70	
9	王涛	￥3,300.00	￥0.00	100	200	￥750.00	￥4,350.00	0	478.5	￥227.50	706	￥3,644.00	
10	罗明	￥3,300.00	￥0.00	140		￥620.00	￥4,060.00		446.6	￥184.00	630.6	￥3,429.40	
11	合计	￥19,100.00	￥1,100.00	900	1050	1370	￥23,520.00	160	2587		3824.7	￥19,695.30	
12	审核:										制表:		

图 26-19　复制公式

从以上过程可看出，在 Excel 中引用加载宏中的函数，与使用 Excel 的内置函数相同。因此，用户可以将常用的自定义函数汇总到一起，创建为加载宏文件，扩充 Excel 的功能。

26.4.2　使用 COM 加载宏

COM 加载项是一个扩展名为 DLL 的文件，DLL 文件需要在 Windows 操作系统中注册，才能使用。如果是使用 26.4.1 节的步骤在 VB6 中生成 DLL 文件，VB 将自动将该 DLL 文件进行注册。若将 COM 加载宏复制到其他计算机中使用，或是从互联网上下载的 COM 加载项，则需要使用手工进行注册。下面介绍使用 COM 加载宏的步骤：

（1）在 Windows 中单击【开始】菜单，选择【运行】命令，打开【运行】对话框。

（2）在【运行】对话框中显示为以下内容，如图 26-20 所示。

```
regsvr32 E:\excel2007 从入门到精通\实例\第 26 章\COM 加载宏\caps.dll
```

提示：DLL 文件放置的位置不同，以上命令后面的参数就不同。

（3）单击【确定】按钮，将弹出如图 26-21 所示提示对话框，完成 DLL 文件的注册。




图 26-20　注册 DLL



图 26-21　注册成功

提示：若要移除或删除 DLL 文件，需先从 Windows 操作系统中将其注销。注册命令只是在上面的 regsvr32 命令中添加参数/u 即可。例如，以下命令可注册上步注册的 DLL 文件。

```
regsvr32 E:\excel2007 从入门到精通\实例\第 26 章\COM 加载宏\caps.dll /u
```

（4）注册 DLL 文件后，就可以向 Excel 中添加 COM 加载宏的引用。启动 Excel，单击【Office 按钮】打开下拉菜单，选择右下方的【Excel 选项】按钮，打开【Excel 选项】对话框。

（5）单击对话框左侧的【加载项】按钮，在对话框下方【管理】标签右侧的下拉列表框中选择【COM 加载项】，如图 26-22 所示。

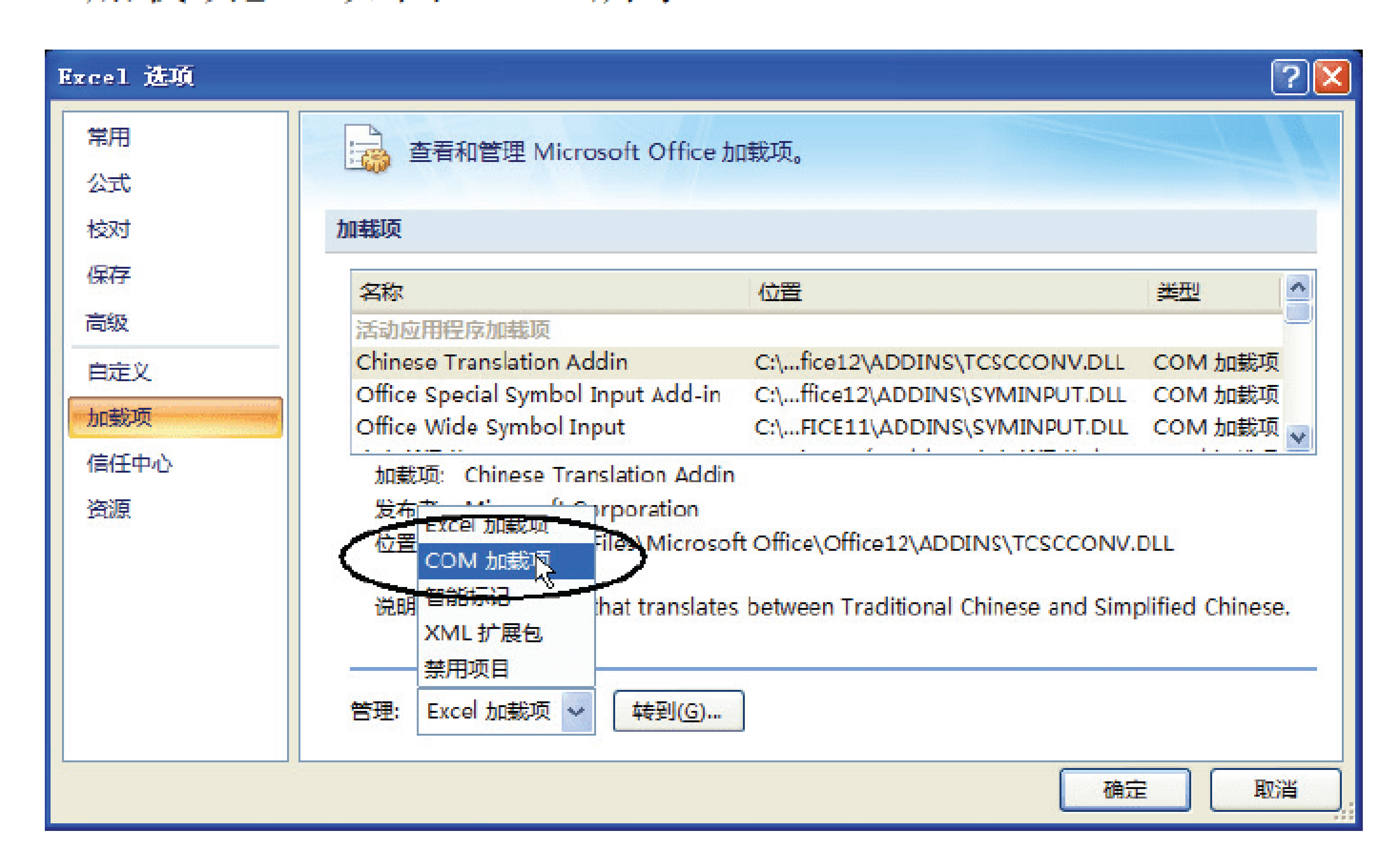


图 26-22　选择 COM 加载宏

（6）在图 26-22 所示对话框中单击右侧的【转到】按钮，打开如图 26-23 所示的【COM 加载项】对话框，在该对话框中显示了 Excel 当前的 COM 加载宏列表。

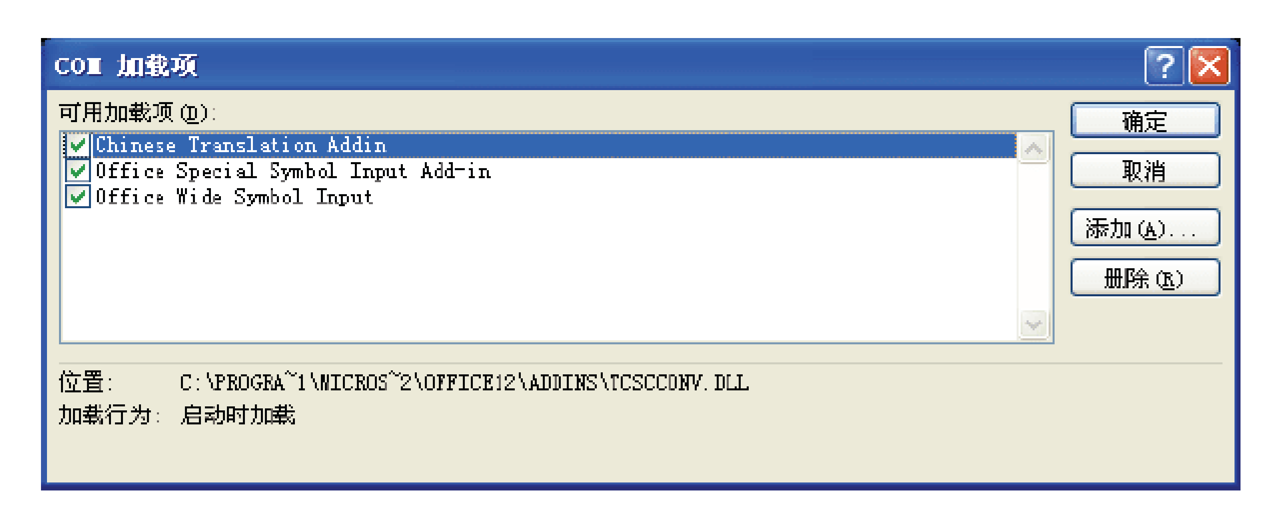


图 26-23　【COM 加载项】对话框

（7）单击【添加】按钮打开【添加加载项】对话框，如图 26-24 所示。

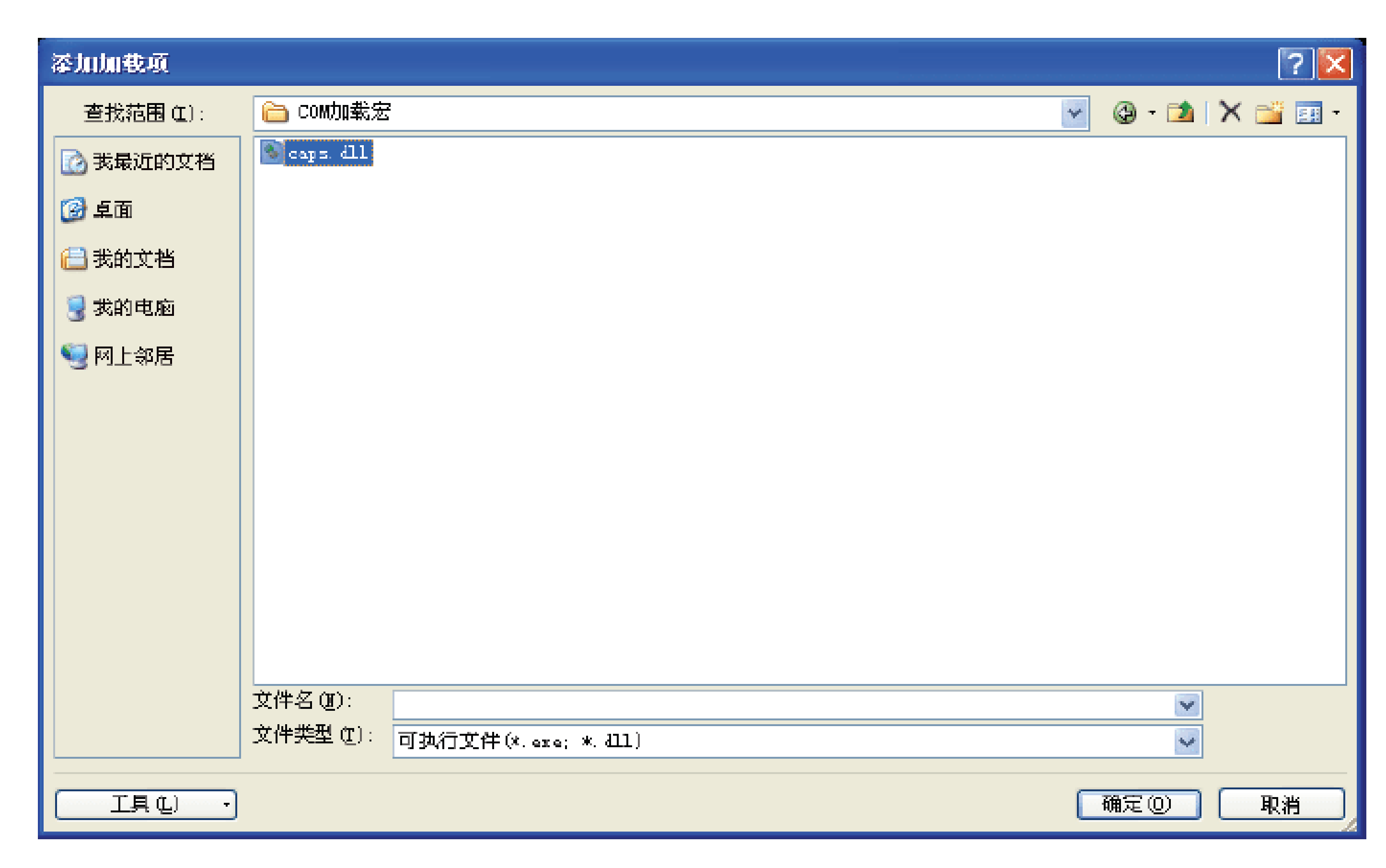


图 26-24　【添加加载项】对话框

（8）在图 26-24 所示对话框中选择 Caps.dll 文件，单击【确定】按钮返回如图 26-25 所示对话框。在该对话框中可看到已将“大小写转换（COM）”宏添加到以下列表中。

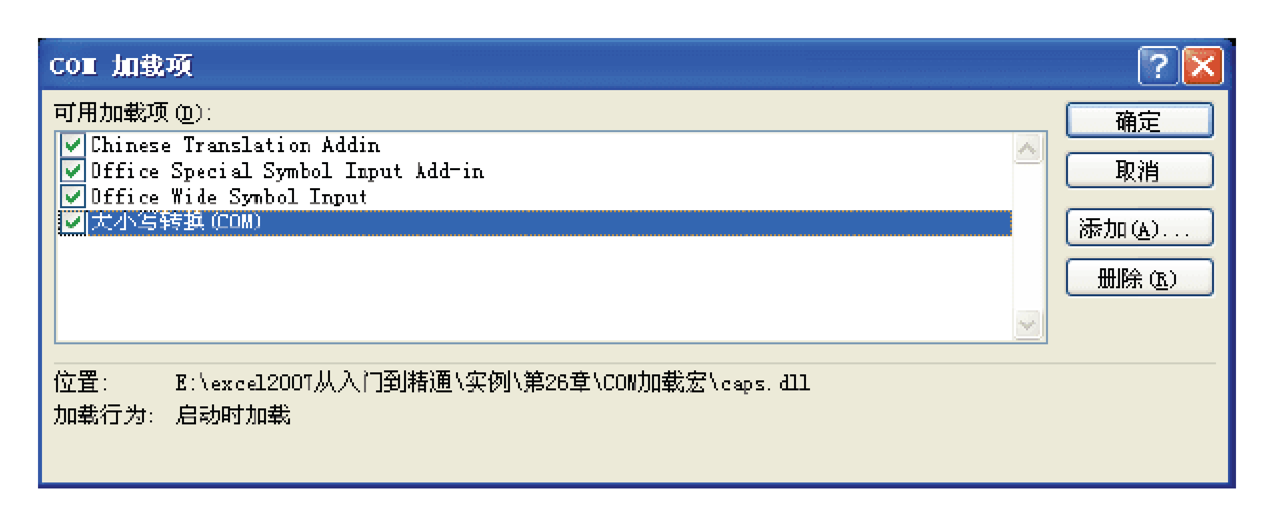


图 26-25　添加自定义 COM 加载宏

（9）加载了“大小写转换(COM)”加载项后，在功能区的【加载项】选项卡中将多出一个按钮，如图 26-26 所示。

（10）在工作表中输入英文单词，并在两个单元格中分别输入单词的小写和大写状态，如图 26-27 所示。

图 26-26　功能区

（11）选择小写单词所在单元格，单击【加载项】选项卡中的【字母大小写转换】按钮，即可将原为小写的单词转换为大写。选择大写单词所在单元格，单击【加载项】选项卡中的【字母大小写转换】按钮，即可将原为大写的单词转换为小写。如图 26-28 所示。

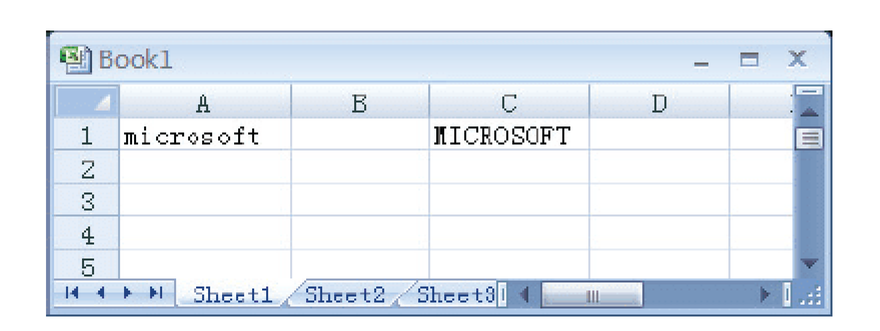


图 26-27　输入单词

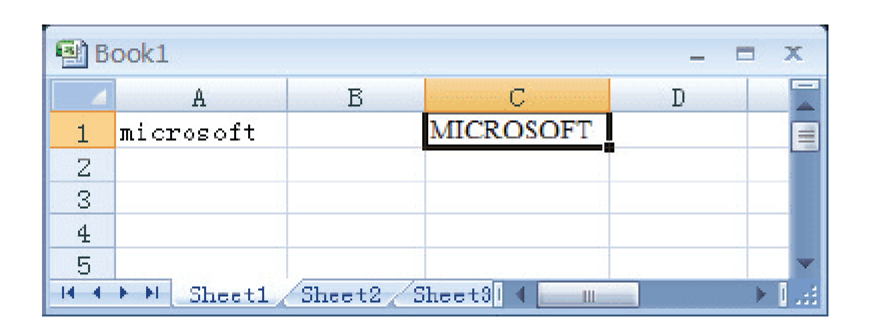


图 26-28　使用 COM 加载宏转换

第 27 章　使用类模块

在 VBA 中，除了模块、用户窗体外，还提供了类模块。类模块向开发人员提供了创建和操作自己的对象类的功能。如果曾使用过 C++之类的面向对象的编写语言，应该知道类的功能和好处。本章将介绍类模块及其作用，并以实例演示在应用程序中使用类模块的方法。

27.1　类模块的概念

在开发 Excel VBA 应用程序时，用户可能从来就不使用类模块。如果作为专业的开发者，将常用代码封装为类模块，可重复应用在不同的应用程序中，从而提高程序的开发效率。

27.1.1　什么是类

简单地说，“类”就是一种类型的对象的表示形式，可以将它想象为描述对象的蓝图。就像一幅蓝图可以用于建成多座建筑一样，一个类也可以用于创建对象的多个副本。

在日常生活中，可以发现相同或相似的物品很多，例如，显示器外壳、鼠标外壳等。在工业化生产中，通常都是先制作一个产品的模具，然后就可通过该模具大量生产相同形状的产品。每一个“模具”就是一个“类”，而生产出来的每一个产品，就是一个“对象”。

在 Excel 中提供非常多的对象，如 Workbook、Worksheet、Range 等，这些对象也都是通过相应的类来创建的。例如：

```
Dim rng1 As Range
```

在以上代码中，Range 是 VBA 内部定义好的一个类（标准类），用户不知道 VBA 是定义该类的代码，但通过 Range 对象提供的属性、方法和事件就可方便地使用该对象。有了标准类 Range，用户在程序中可使用该类反复定义多个具有相同类的对象，如：

```
Dim rng2 As Range
Dim rng3 As Range
Dim rng14 As Range
```

除了使用 Excel 提供的标准类外，VBA 还提供类模块功能，使开发者可根据需要定义自己的类。

在自定义类中，通过属性、方法和事件向用户提供服务，把完成运算的过程隐藏起来。

27.1.2　类的作用

在 VBA 中，开发人员不使用类模块也能开发绝大多数的应用系统。不使用类模块编写程序时，程序代码和程序处理的数据是分离的。这样可能会出现多个不同过程用一定的规则修改数据。当需要调整修改数据的规则时，就需要修改多段程序的代码。

如果使用类将数据和处理代码封装在一起，将使管理任务更加简单。主要体现在以下几个方面：

- 数据作为对象的属性，都与特定对象相关联，使数据与对象具有特定的关系（不再是分离的松散关系）。
- 数据作为对象的属性，用户只能通过设置的接口进行修改，使对象的数据可保持在有效的范围之内。
- 使用对象提供的方法，对一个对象可以进行的操作提供统一接口。在对象的定义中，只需要修改一次代码，即可调整数据的修改规则。
- 使用类的另一个好处就是可以将复杂的过程封装在类的内部，类的使用人员不需要知道类的实现过程，只需要熟悉类的属性、方法和事件即可。

在一个大型系统开发过程中，各部门将自身的业务流程编写封装到类中。其他部门使用该类时，对于实现的技术细节不用关心，只需要了解其接口（属性、方法和事件）即可。

27.1.3　理解类

通常，所有 VBA 的语句、函数以及标准类等资源都可以在自定义类中使用。创建自定义类时，对于变量、过程和函数等概念，在自定义类中又有不同的意义。

1．理解对象

对象是由类创建的一个实例，它是类的实体化。对象的引用和操作被划分为 3 个部分：

- 属性是指对象的特性。如 Worksheet 对象，有包含单元格的行数、列数等属性。
- 方法是指对象的某个操作。如 Worksheet 对象，有增加工作表、删除工作表等方法。
- 事件是指对象对外部动作的响应。例如，用鼠标单击 Worksheet 对象的单元格时，会产生一个 Change 事件，修改单元格内容时，会产生一个 SelectionChange 事件。

2．变量的作用域

在本书前面相关章节介绍了变量的作用域。变量可以划分为过程级、模块级和全局变量。在类模块中，对变量作用域的理解要注意以下两点：

- 由于类是生成对象的模板，每一个对象，相当于是类的一个副本，对象之间是相互独立的，因此，模块级的变量只作用于对象自身，对其他通过该类创建的对象不会起作用。
- 在类模块中使用 Public 关键字声明的变量，通过该类生成的对象都可访问。

3．过程和函数

变量、过程、函数是标准模块中使用的最基本的构件，在类模块中，它们仍然是最基本和重要的角色。过程和函数并无实质的区别，当需要返回值时，就使用 Function，如果不需要返回任何结果，可使用 Function，也可使用 Sub。

过程（Sub）、函数（Function）也有作用域，在标准模块中通过使用 Private 和 Public 关键字（可以省略 Public 关键字，因为它是默认的），可以划分为模块级和全局级，以决定它是在当前的模块内有效还是在整个工程内有效。

同变量一样，在类模块中使用 Public 关键字的函数或过程，才能被声明的对象访问。使用 Private 关键字的函数或过程只能在类模块中进行调用（不对用户开放）。

4．通用内部控件

VBA 提供了一个名为 Control 的类，可用来引用一般的内部控件。例如，使用以下代码声明一个控件对象：

```
Dim  cnt  As  Control
```

就可以将任何控件赋给该变量进行保存，而不管具体的类型。因为在自定义类中通常要处理大量相近的对象，所以这种特性非常有用。在实际使用时，可以通过容器控件的 Controls 属性来返回一个 Control 的集合对象。

例如，使用以下代码可以在当前用户窗体中查询指定名称的按钮：

```
Dim cnt As Control
For Each cnt In Me.Controls
   If TypeName(cnt) = "CommandButton" Then MsgBox cnt.Caption
Next
```

5．集合Collection

Collection 是在使用类时最常用到的对象。一个 Collection 对象代表一组相关的项目（在 Collection 对象中也可保存不同类型的数据，但保存不同类型数据时不利用程序处理）。例如，使用以下代码可创建一个集合对象：

```
Dim col As New Collection
```

集合建立后，可以使用 Add 方法添加成员，用 Remove 方法删除成员，用 Item 方法从集合中返回特定成员。

27.2　创建类模块

本节用一个实例演示创建类模块的方法。本节创建一个“数据库类”，通过该类可创建与 Access 数据库的连接。将数据库连接字符串封装在类模块中，用户通过属性给出要访

问的 Access 数据库名、SQL 语句，执行内置的方法即可返回记录集。

27.2.1　建立对象类

要创建自定义类，首先必须先新建一个类模块，具体步骤如下：

（1）在 Excel 中按 Alt+F11 切换到 VBE 环境中。

（2）在 VBE 中单击主菜单【插入】|【类模块】命令，向工程插入一个类模块。

（3）在【属性】窗口中修改类的【(名称)】为 clsADO，如图 27-1 所示。

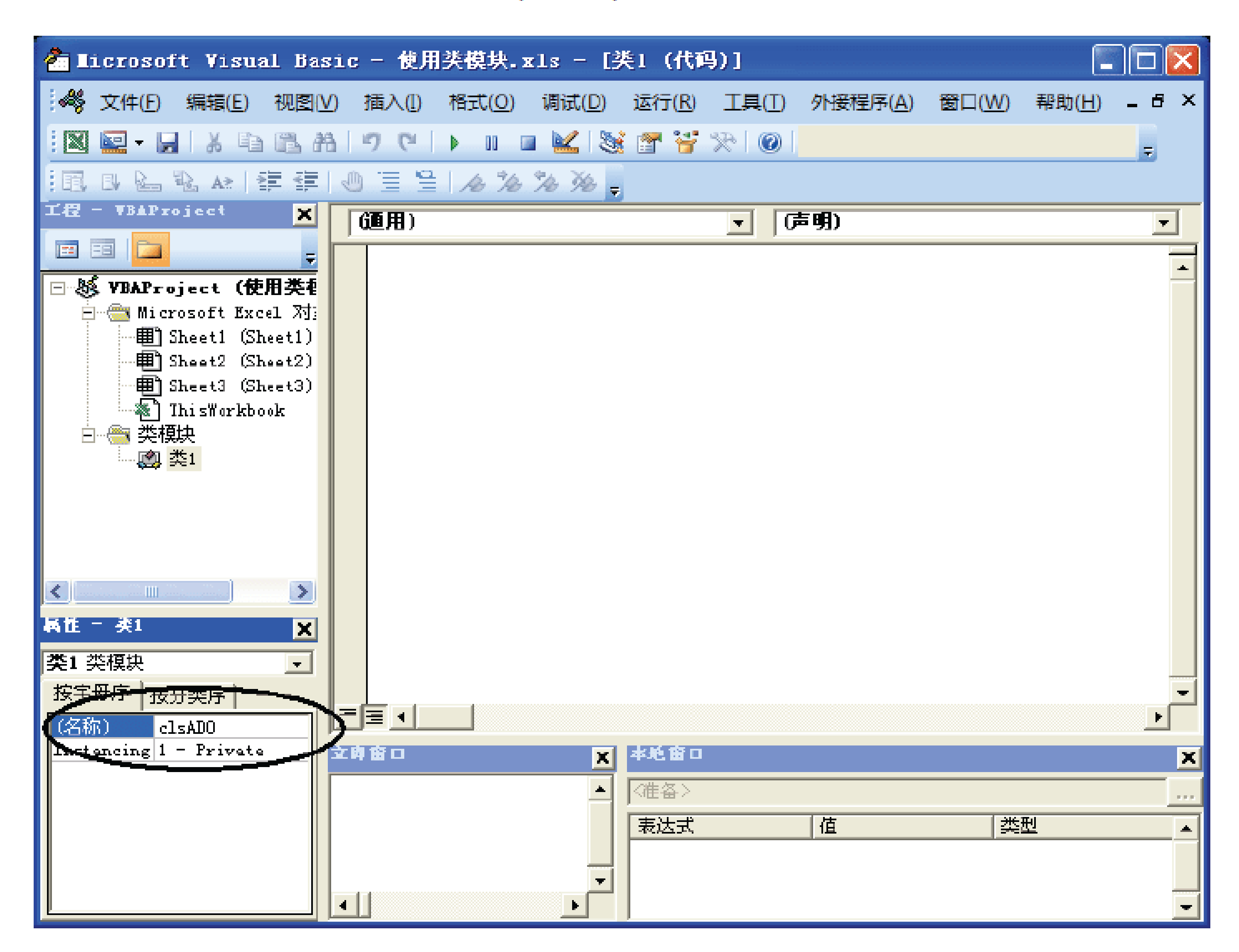


图 27-1　插入类模块

注意：所有 VBA 类模块都有【(名称)】属性，用来确定类的名称，最好为其定义一个易于理解的类名称。每一个类模块只能定义一个对象类。

27.2.2　建立类的属性

通过属性保存类中的数据，在 VBA 中有两种建立属性的方法：

- 在类模块的声明部分定义的 Public 变量可被对象外部的代码访问，可作为对象的属性。这种方法的缺点是：类不能知道属性值何时被外部过程修改了，也就无法将属性值限定在一个有效的范围内。
- 使用 Property 过程来定义属性值。当外部过程修改属性值时，将执行 Property 过程，在该过程中可编写代码，对设置的值进行检查，控制其在一个规定的范围内。

使用 Property 过程可对一类模块添加和操作属性，Property 过程以 Property Let、Property Get 或 Property Set 语句开始，而以 End Property 语句退出。这 3 种形式的 Property 过程的作用分别如下所述。

- Property Let：这类过程用来设置类模块的属性值。
- Property Get：这类过程用来读取类模块的属性值。
- Property Set：这类过程用来设置对对象的引用。

在使用 Property 过程设置属性值时，类模块中的所有变量可声明为 Private 类型，以避免过程直接修改类模块中变量的值。

创建属性时一般按以下步骤进行：

（1）创建私有变量。在创建 clsADO 类时，需要在类模块的声明部分使用以下代码声明变量。

```
Option Explicit
'使用本类模块，需添 ADO 的引用"Microsoft ActiveX Data Objects 2.8 Library"
Private cnn As Connection             '连接对象
Private rs As Recordset               '记录集对象
Private m_DBFullName As String        'Access 数据库名称（包含路径信息）
Private m_CommandText As String       'SQL 语句
```

提示：习惯上，一般类的私有成员变量前面加上前缀“m_”。

（2）创建 Property Get 过程来获取对象的属性值。

（3）创建 Property Let 过程来设置上一步定义的私有变量的值。

27.2.3 创建 Property Get 过程

Property Get 语句用来获取对象的属性。其基本形式就是一个声明和一个主体。声明包括属性名和数据类型。

每个属性值都需要创建一个 Property Get 过程。对于 clsADO 类，其 2 个属性值的 Property Get 过程代码如下：

```
Property Get DBFullName() As String    '返回数据库名称
    DBFullName = m_DBFullName
End Property

Property Get CommandText() As String   '返回 SQL 语句
    CommandText = m_CommandText
End Property
```

从以上代码可以看出，在每个 Property Get 的过程声明行中都指定了属性的名称和数据类型，如：

```
Property Get DBFullName() As String
```

定义属性名为 DBFullName，该属性的数据类型为 String。

Property Get 过程和函数过程非常相似，给过程名赋值就可返回值。在 Property Get 过程内部，一般为一条赋值语句：

```
DBFullName = m_DBFullName
```

其中 DBFullName 为属性名称，而 m_DBFullName 是类的私有数据变量。也可将一个运算结果赋值给属性值。

27.2.4　创建 Property Let 过程

与 Property Get 过程相对应的就是 Property Let 过程。通过 Property Let 过程可让使用者改变对象的属性值。

一般对每个属性值都需要编写一段 Property Let 过程。如果对象的某个属性为只读，则不需要定义 Property Let 过程。

```
Property Let DBFullName(strDBFullName As String)   '设置数据名称
    If LCase(Right(strDBFullName, 4)) <> ".mdb" And _
        LCase(Right(strDBFullName, 6)) <> ".accdb" Then
        Exit Property
    End If
    m_DBFullName = strDBFullName
End Property

Property Let CommandText(strSQL As String)          '设置 SQL 语句
    m_CommandText = strSQL
End Property
```

从以上代码可以看出，Property Let 过程至少需要一个参数来设置属性值。与 Property Get 过程相比，在 Property Let 过程中可能还需要做一些检查工作，因为外部过程有可能会为属性设置一个超过范围的数据。例如设置数据库名称时，可使用 If 语句对传入的文件进行判断，若右侧字符不是 Access 数据库扩展名（扩展名为.mb 或 cccdb），则退出过程。

27.2.5　创建类的方法

除了属性之外，一般对象都有至少一个方法。方法是对象可以执行的操作，使用方法可以操作对象类中的数据。

在 clsADO 类中创建了多个方法，各方法完成的任务如下所述。

- ConnDB 方法：创建数据库连接实例。
- CloseDB 方法：关闭数据库连接实例。
- RunSQL 方法：获取记录集对象。

1. ConnDB方法

ConnDB 方法用来创建数据库的连接，该方法将使用私有成员变量 m_DBFullName。具体代码如下：

```
Private Sub ConnDB()    '连接数据库
    If cnn.State = 0 And Len(m_DBFullName) > 0 Then
        cnn.Provider = "Microsoft.Jet.OLEDB.4.0"
        cnn.Open m_DBFullName
```

```
    End If
End Sub
```

2. CloseDB方法

CloseDB 方法用来关闭数据库连接，具体代码如下：

```
Public Sub CloseDB()    '关闭连接
    cnn.Close
End Sub
```

注意：CloseDB 方法中不使用 Set cnn=Nothing 语句，因为使用该语句后，变量将从内存中释放。这时如果重新设置自定义类 clsADO 的属性后，将无法再使用 ConnDB 方法，因为变量 cnn 已经被释放。

3. RunSQL方法

RunSQL 方法将使用私有变量 m_CommandText 中设置的 SQL 语句，根据该 SQL 语句返回一个记录集。具体代码如下：

```
Public Function RunSQL() As Recordset       '执行 SQL 语句
    If Len(m_CommandText) > 0 Then
        Call ConnDB                         '创建数据库连接
        If rs.State <> 0 Then rs.Close
        rs.Open Source:=m_CommandText, _
                ActiveConnection:=cnn, _
                CursorType:=adOpenStatic, _
                LockType:=adLockReadOnly
        Set RunSQL = rs                     '返回记录集
    End If
End Function
```

在该函数的过程声明行中，定义了方法执行后的返回结果类型。另外，使用 Public 声明方法，使其在类模块之外也可访问。

27.2.6 类模块的事件

事件是对象可识别的动作，如按钮对象可识别 Click 事件。自定义类模块有 Initialize 事件和 Terminate 事件，这两个事件分别在类的实例初次创建时和最后一个指针释放或破坏时触发。可以用 Initialize 事件设置对象类的默认属性值，用 Terminate 事件进行销毁对象前的整理工作。

在 clsADO 类中，在 Initialize 事件中编写代码，实例化数据库连接对象和记录集对象。编写 Initialize 事件代码的过程如下：

（1）在类模块代码窗口的对象下拉列表中选择 Class，在事件下拉列表中选择 Initialize，将在代码窗口中生成该事件过程的结果。

（2）在 Initialize 事件过程中输入以下代码：

```
Private Sub Class_Initialize()  '初始化类
    Set cnn = New Connection
    Set rs = New Recordset
End Sub
```

（3）用类似的方法，为类模块的 Terminate 事件过程中编写代码，用来清理终止类时需进行的工作。

```
Private Sub Class_Terminate()   '终止类
    If cnn.State = 1 Then
        cnn.Close
    End If
    Set cnn = Nothing
    Set rs = Nothing
End Sub
```

27.3　使用类模块创建对象

在类模块中编写好类模块代码后，就可以在 VBA 的模块中使用该类模块来定义对象，通过设置类模块的属性、调用类模块的方法来使用类模块的功能。

下面演示使用上节创建的类模块 clsADO 的方法，具体步骤如下：

（1）在 VBE 中单击主菜单【插入】|【模块】命令，向工程中增加一个模块。

（2）在模块的声明部分输入代码，声明变量为类模块。如图 27-2 所示的列表框中，将显示 27.2.6 节定义的类模块名称。

（3）使用类模块定义变量后，在代码窗口输入对象名即可看到该对象提供的方法和属性，如图 27-3 所示。

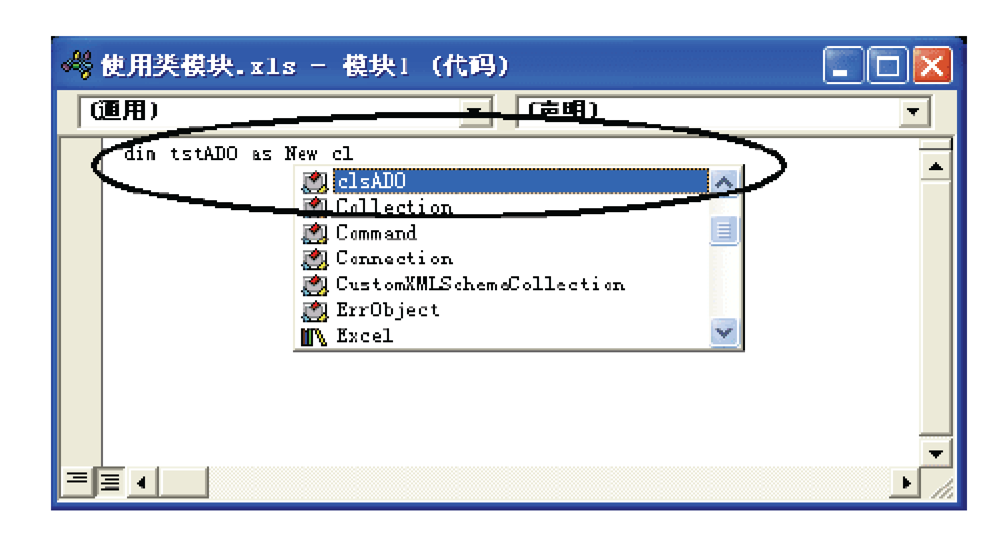



图 27-2　用类模块声明变量

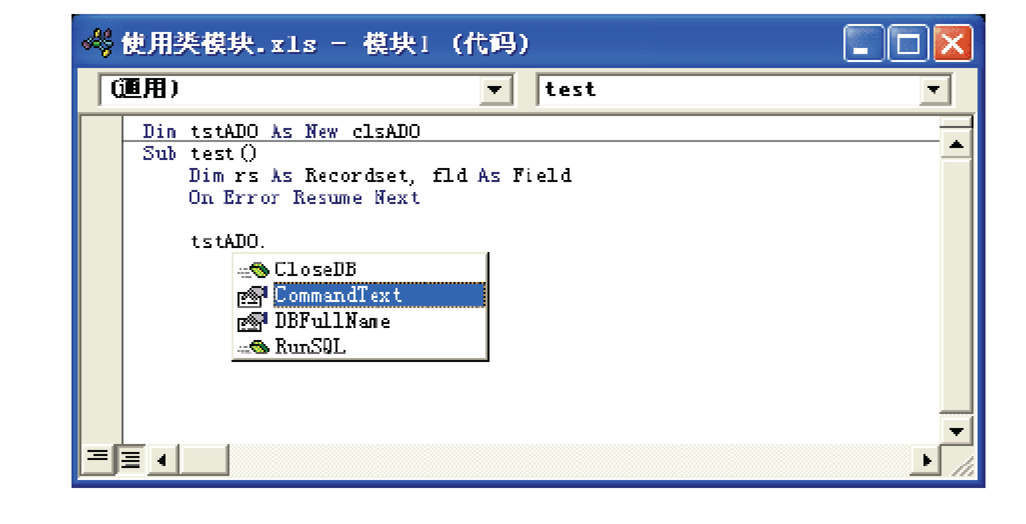


图 27-3　类模块提供的属性和方法列表

（4）接着编写以下过程，设置对象 tstADO 的 DBFullName 属性和 CommandText 属性，调用 tstADO 对象的 RunSQL 方法获取记录集对象。最后对该记录集进行处理，得到相应的数据。

```
Sub test()
    Dim rs As Recordset, fld As Field
```

```
    On Error Resume Next

    tstADO.DBFullName = ThisWorkbook.Path & "\Northwind.mdb"    '设置属性
    tstADO.CommandText = "select * from 客户"               '设置属性

    Set rs = tstADO.RunSQL                                  '调用自定义对象的方法
    With Worksheets("Sheet1")
        i = 1: j = 1
        For Each fld In rs.Fields
            .Cells(i, j) = fld.Name
            j = j + 1
        Next
        j = 1
        i = i + 1
        Do While Not rs.EOF
            For Each fld In rs.Fields
                .Cells(i, j) = fld.Value
                j = j + 1
            Next
            j = 1
            rs.MoveNext
            i = i + 1
        Loop
    End With
End Sub
```

执行以上过程，将在工作表 Sheet1 中显示记录集中的数据，如图 27-4 所示。

使用类模块.xls [兼容模式]

	A	B	C	D	E	F	G	H
1	客户ID	公司名称	联系人姓名	联系人职务	地址	城市	地区	邮政编码
2	ALFKI	三川实业有	刘小姐	销售代表	大崇明路	天津	华北	3435
3	ANATR	东南实业	王先生	物主	承德西路	天津	华北	2345
4	ANTON	坦森行贸易	王炫皓	物主	黄台北路	石家庄	华北	9850
5	AROUT	国顶有限公	方先生	销售代表	天府东街	深圳	华南	8908
6	BERGS	通恒机械	黄小姐	采购员	东园西甲	南京	华东	7980
7	BLAUS	森通	王先生	销售代表	常保阁东	天津	华北	7870
8	BLONP	(东北)国皓	黄雅玲	市场经理	广发北路	大连	东北	5654
9	BOLID	迈多贸易	陈先生	物主	临翠大街	西安	西北	9079
10	BONAP	祥通	刘先生	物主	花园东街	重庆	西南	5676
11	BOTTM	广通	王先生	结算经理	平谷嘉石大	重庆	西南	8080
12	BSBEV	光明杂志	谢丽秋	销售代表	黄石路 50	深圳	华南	7609
13	CACTU	(东北)威航	刘先生	销售代理	经七纬二路	大连	东北	1204
14	CENTC	(东北)三捷	王先生	市场经理	英雄山路	大连	东北	1300

Sheet1 Sheet2 Sheet3

图 27-4　测试效果

第 28 章　操作 VBE

Excel 开发人员通过 VBE 可编写 VBA 代码、制作用户窗体。其实，VBE 也包含一个对象模型，通过该对象模型可控制 VBA 工程的主要元素，可以编写 VBA 代码添加或删除模块、创建用户窗体、生成 VBA 代码。

28.1　VBE 简介

在本书第 3 章中介绍了 VBE 操作环境的使用方法。其实在 VBE 开发环境中，VBE 本身也是一个对象模型。该对象模型展示了 VBA 工程的主要元素，通过该对象模型，用户可编写 VBA 代码来添加或删除模块、生成其他的 VBA 代码、创建用户窗体等操作。

28.1.1　添加 VBE 对象模型的引用

要在 VBA 工程中使用 VBE 对象模型，必须先将 VBE 对象模型库添加到当前工程中。具体步骤如下：

（1）在 VBE 开发环境中单击主菜单【工具】|【引用】命令，打开【引用】对话框，如图 28-1 所示。

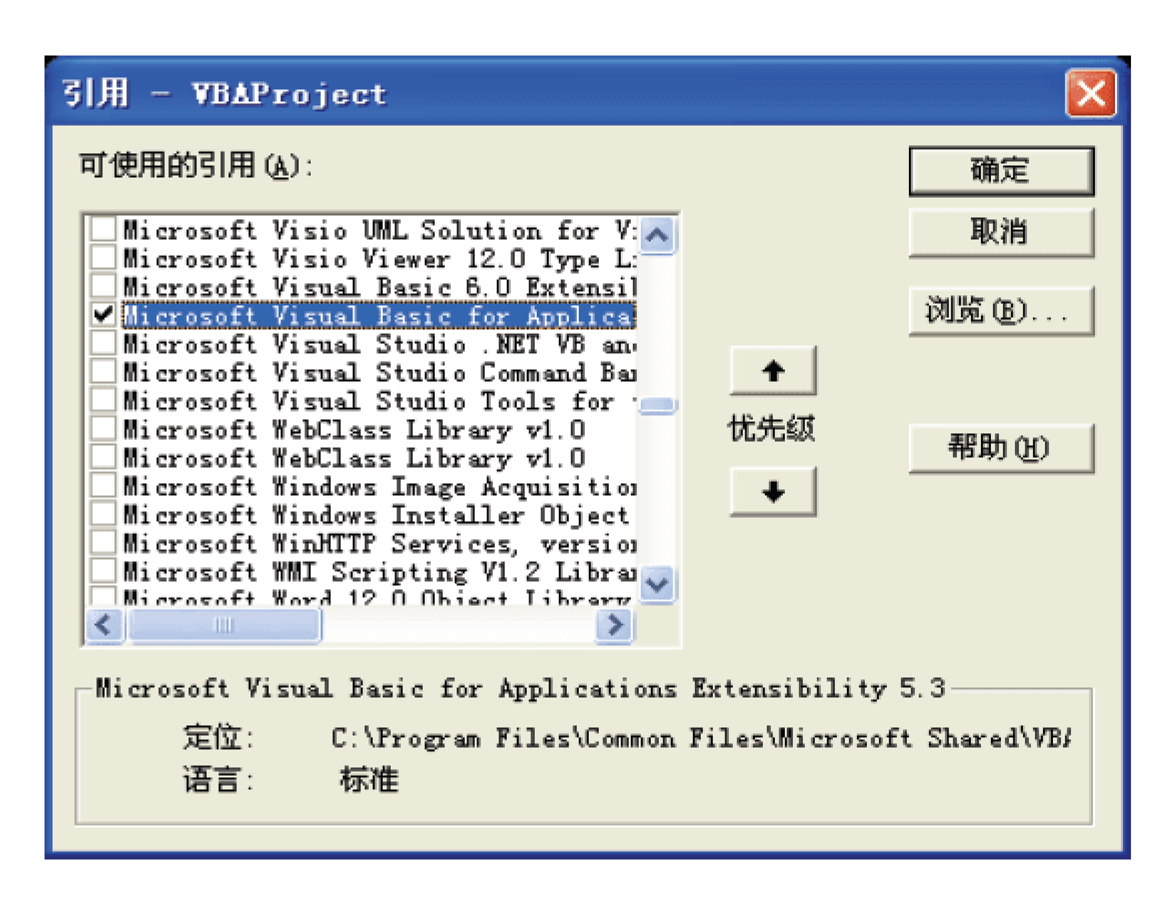


图 28-1　添加 VBE 引用

（2）在对话框中选择 Microsoft Visual Basic for Applications Extensibility Library 项，单击【确定】按钮将其添加到当前工程中。

28.1.2　信任 VBA 访问 VBE 对象模型

为了保证 VBA 代码的安全，在默认情况下，是不允许用户使用代码访问 VBE 对象模型的。若代码访问 VBE 对象模型，将弹出如图 28-2 所示的警告提示框。

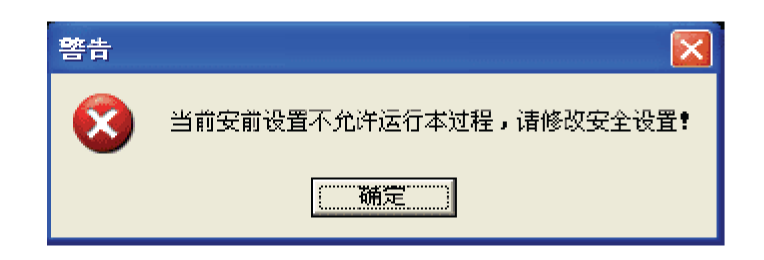


图 28-2　警告信息

要在 VBA 代码中访问 VBE 对象模型，需要对宏安全进行设置。具体步骤如下：

（1）单击【Office 按钮】打开下拉菜单，单击右下角的【Excel 选项】按钮打开【Excel 选项】对话框。

（2）单击对话框左侧的【信任中心】按钮显示信任中心页面，如图 28-3 所示。

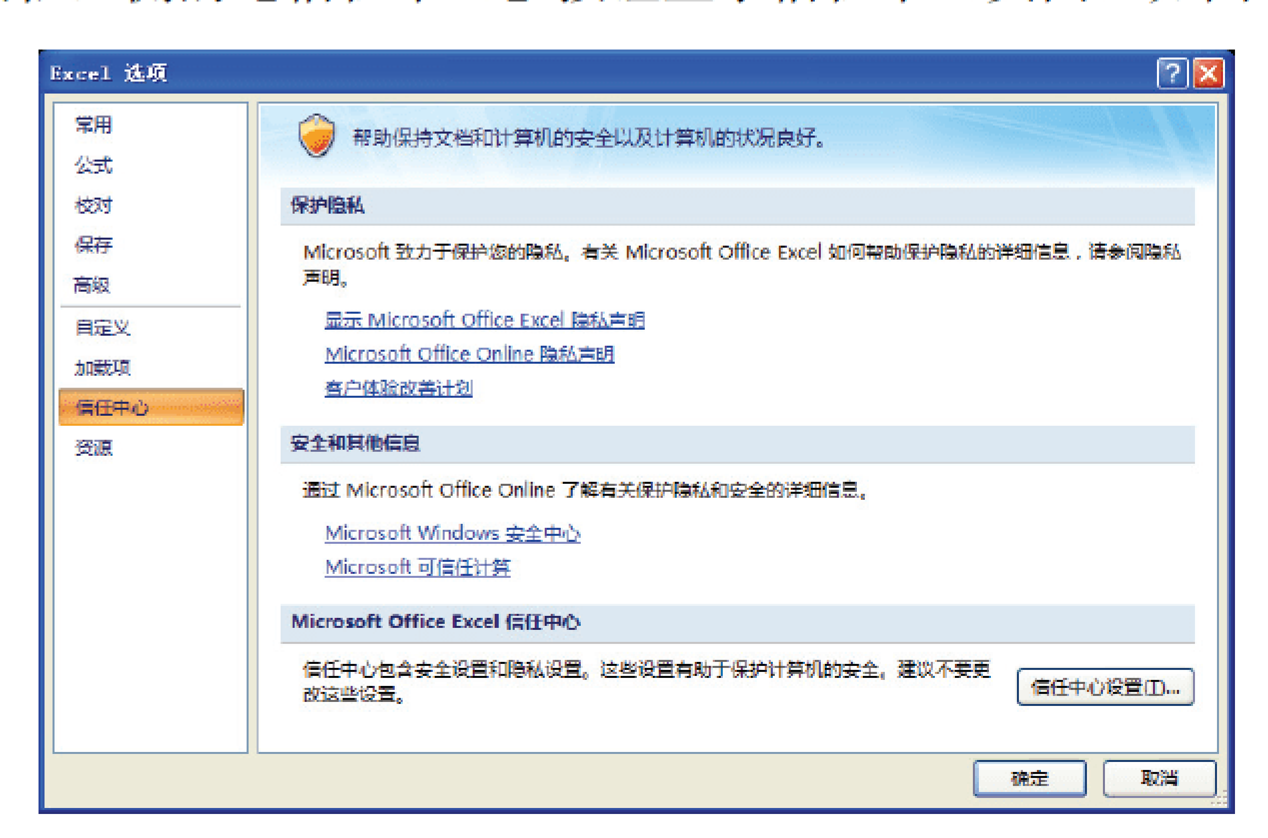


图 28-3　【信任中心】选项卡

（3）在图 28-3 所示对话框中单击【信任中心设置】按钮，打开如图 28-4 所示的界面。在宏的安全设置中选中【信任对 VBA 工程对象模型的访问】单选框。

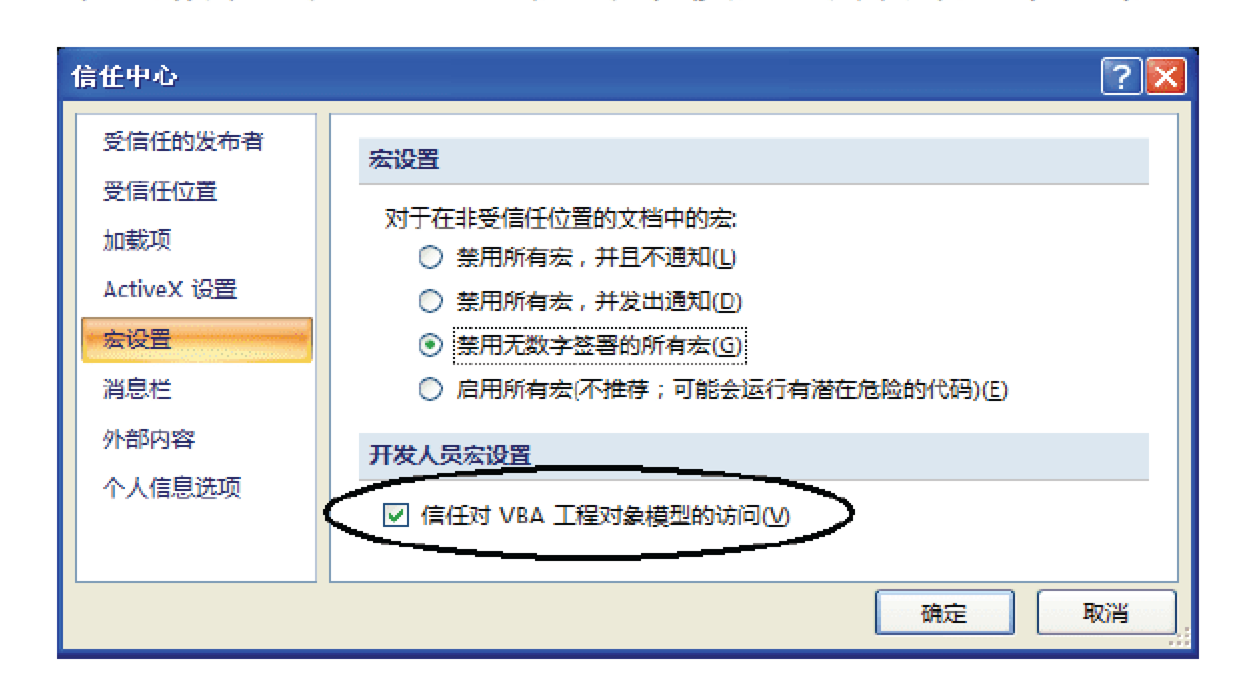


图 28-4　信任对工程模型的访问

（4）单击【确定】按钮，完成设置。

28.2　VBE 对象模型

VBE 的编程模型也是面向对象结构的，对 VBE 进行程序设计需先了解其对象模型。本节将简单介绍常用对象。

28.2.1　了解 VBE 对象模型

VBE 对象模型如图 28-5 所示。每个打开的工作簿或加载宏都有一个 VBProject 对象代码，每个工作簿中包含多个 VBComponent 对象（Excel 对象、窗体、模块和类模块），而 Reference 对象则表示工作引用的外部模型库，每个外部模型库为一个 Reference 对象。

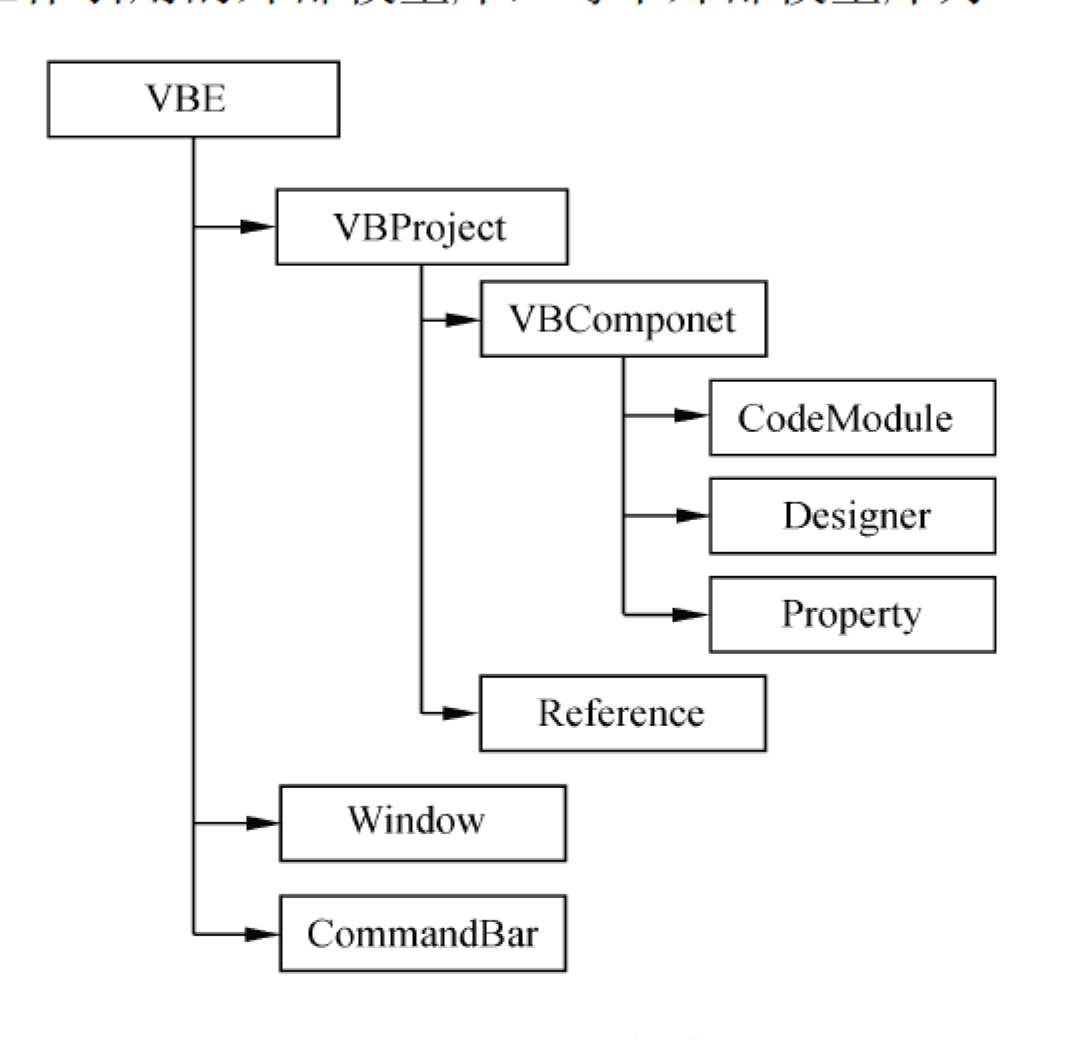


图 28-5　VBE 对象模型

28.2.2　VBProject 对象

VBProject 对象表示一个工程。可用 VBProject 对象设置工程的属性、访问 VBComponents 集合以及访问 References 集合。

VBProjects 集合表示开发环境中所有打开的工程。在开发环境的实例中可用 VBProjects 集合访问具体的工程。VBProjects 是一个可用于 For Each 块的标准集合。

使用 Workbook 对象的 VBProject 属性，可返回一个 VBProject 对象，代表该工作簿的工程。通过 VBProject 对象的属性可对工程进行设置，常用的属性包括以下几种。

- Protection 属性：返回一个值，指示工程的保护状态，此属性为只读。返回常数 Vbext_locked 表示指定的工程是被锁住的，返回常数 Vbext_pp_none 表示指定的工

程并没有被保护。

- ❑ Name 属性：返回或设置活动的工程名称。
- ❑ Mode 属性：返回工程当前所处的方式，可为运行方式（vbext_vm_Run）、中断方式（vbext_vm_Break）、设计方式（vbext_vm_Design）。
- ❑ VBE 属性：返回该 VBE 对象的根。所有的对象都有一个指向 VBE 对象根的 VBE 属性。
- ❑ Saved 属性：指示工程自上一次保存后是否编辑过。

28.2.3 VBComponent 对象

VBComponent 对象代表一个包含在工程中的部件，例如类模块或标准模块。使用 VBComponent 对象访问与部件关联的代码模块或改变部件的属性设置。

可以使用 Type 属性找出 VBComponent 对象所引用的部件类型，使用 Collection 属性找出该部件在哪个集合中。

VBComponents 集合表示工程中所有的部件。使用 VBComponents 集合在工程中访问、添加和删除部件。部件可以是窗体、模块、或类。

VBComponent 对象的常用属性和方法如下面所述。

❑ CodeModule 属性：返回一个对象，代表与部件相关的代码。如果该部件没有关联的代码模块，则 CodeModule 属性返回 Nothing。

❑ Type 属性：返回一个数字或字符串值，表示 VBComponent 对象代表的部件类型，可为以下几种类型。

  - ➢ Vbext_ct_ClassModule：类模块。
  - ➢ Vbext_ct_MSForm Microsoft：窗体。
  - ➢ Vbext_ct_StdModule：标准模块。

- ❑ Name 属性：返回或设置在代码中用来识别部件的名称，如果试图将 Name 属性设置成一个已经被使用或无效的名称，将生成一个错误。
- ❑ Export 方法：使用该方法将部件按单独文件或一些文件进行保存。

28.2.4 Reference 对象

Reference 对象表示类型库或工程的引用。如果引用不再指向有效的引用，则 IsBroken 属性返回 True。如果引用是一不能移动或删除的默认引用，则 BuiltIn 属性返回 True。可用 Name 属性决定所要添加或删除的引用是否正确。

Reference 对象的常用属性如下面所述。

- ❑ BuiltIn 属性：指出该引用是否不能被删除的默认引用。
- ❑ Description 属性：返回或设置一个字符串表达式，表示引用的描述性的名称。
- ❑ FullPath 属性：返回引用的类型库的路径和文件名。
- ❑ IsBroken 属性：指定 Reference 对象是否指向一个注册表中有效的引用。如果返回为 True，则表示 Reference 对象不再指向一个在注册表中的有效引用。

- ❑ Major 属性：返回被引用的类型库的主版本号。
- ❑ Minor 属性：返回被引用的类型库的次版本号。
- ❑ Name 属性：返回代码中引用的名称。
- ❑ Type 属性：返回一个数字或字符串值，指示 Reference 对象的类型。可返回常数 vbext_rt_TypeLib（表示类型库）和 vbext_rt_Project（表示工程）。

28.2.5　CodeModule 对象

CodeModule 对象表示在诸如窗体，类或文档等部件中的程序代码。

可用 CodeModule 对象来修改（添加、删除、编辑）与部件相关联的代码。每个部件都与一个 CodeModule 对象相关联。但是，一个 CodeModule 对象可以与多个代码窗格相关联。

与 CodeModule 对象相关联的方法，能让开发人员操作并返回有关逐行代码文本的信息。例如，可用 AddFromString 方法将文本添加到模块中。AddFromString 将文本放在第一个过程之上，如果没有过程，则将文本放在模块尾端。

1. 常用属性

（1）CountOfDeclarationLines 属性

该属性返回代码模块的【声明】部分的代码行数。

（2）CountOfLines 属性

该属性返回代码模块中代码的行数。

（3）ProcBodyLine 属性

该属性返回过程的第一行。其语法格式如下：

```
object.ProcBodyLine(procname, prockind) As Long
```

参数的含义如下所述。

- ❑ procname：表示过程名。
- ❑ prockind：指定要定位的过程种类。

提示：Sub、Function 或 Property 出现在一个过程的第一行。

（4）ProcCountLines 属性

该属性返回特定过程的行数。其语法格式如下：

```
object.ProcCountLines(procname, prockind) As Long
```

ProcCountLines 属性返回在过程声明之前的所有空行及注释行的计数，并且，如果该过程是一段代码模块的最后一个，那么此过程之后的所有空行也计入。

（5）ProcOfLine 属性

该属性返回特定的行所在的过程名。其语法格式如下：

```
object.ProcOfLine(line, prockind) As String
```

参数 line 设置要检查的行数。参数 prockind 指定要定位的过程种类。

一个过程声明之前的空行或注释行属于该行，并且，如果该过程是一个代码模块的最后一个过程，则该模块后面将有一行或多行空白行。

ProcCountLines 属性返回在过程声明之前的所有空行及注释行的行数，并且，如果该过程是一段代码模块的最后一个，那么紧接着该过程之后的空行也计算在内。

（6）Lines 属性

该属性返回一个字符串，包含指定行数的代码。其语法格式如下：

```
object.Lines(startline, count) As String
```

参数 startline 指定起始行号，count 指定要返回代码的行数。

提示：代码模块中的行号是从 1 开始。

2. 常用方法

（1）AddFromFile 方法

该方法添加文件内容到模块中。其语法格式如下：

```
object.AddFromFile(filename)
```

参数 Filename 用来指定欲添加到工程或模块的文件名。

AddFromFile 方法可在代码模块中第一个过程之前的行开始插入文件的内容。如果模块没有包含过程，AddFromString 可将文件的内容放置在模块的最后。

（2）ReplaceLine 方法

该方法用特定的代码代替原代码。其语法格式如下：

```
object.ReplaceLine(line, code)
```

参数 Line 用来指定所要代替的行。Code 用来指定要插入的代码。

（3）InsertLines 方法

该方法在一个代码块的某个指定位置插入一行或多行的代码。其语法格式如下：

```
object.InsertLines(line, code)
```

如果使用 InsertLines 方法所插入的文本用换行回车分隔，则将依此插入一些行。

（4）DeleteLines 方法

该方法删除一个单行或指定行范围的代码。其语法格式如下：

```
object.DeleteLines (startline) [count]
```

参数 Startline 用来指定欲删除的开始行。Count 用来指定欲删除的行数。

如果没有指定欲删除的行数，DeleteLines 将只删除一行。

28.3 显示 VBA 工程相关信息

通过 VBE 对象模型，使用 28.2.5 节介绍的各子对象提供的属性和方法，可查看工程

的相关信息。

28.3.1　查看工程信息

通过 VBProject 对象的属性，可获取工程的信息。例如，以下代码将显示当前工作簿的工程信息。

```
Sub 显示工程信息()
    Dim oVBP As VBIDE.VBProject
    On Error Resume Next
    Set oVBP = ActiveWorkbook.VBProject
    If Err <> 0 Then
        MsgBox "当前安全设置不允许运行本过程，请修改安全设置！", _
            vbCritical + vbOKOnly, "警告"
        On Error GoTo 0
        Exit Sub
    End If
    arr1 = Array("工程名称", "完整路径", "描述", "工程类型", "帮助文件", "保护状
态", "运行状态")
    With Worksheets("工程信息")
        .Columns("1:2").Clear
        .Range("A2:A8") = WorksheetFunction.Transpose(arr1)
        .Cells(2, 2) = oVBP.Name          '工程名称
        .Cells(3, 2) = oVBP.Filename      '完整路径
        .Cells(4, 2) = oVBP.Description '描述
        .Cells(5, 2) = oVBP.Type          '工程类型
        .Cells(6, 2) = oVBP.HelpFile      '帮助文件
        .Cells(7, 2) = IIf(oVBP.Protection = vbext_pp_locked, "保护", "未保
护")
        Select Case oVBP.Mode             '工程状态
           Case 0: .Cells(8, 2) = "运行状态"
           Case 1: .Cells(8, 2) = "中断状态"
           Case 2: .Cells(8, 2) = "设计状态"
        End Select
    End With
End Sub
```

执行以上代码，将在工作表“工程信息”中列出当前工程的信息，如图 28-6 所示。

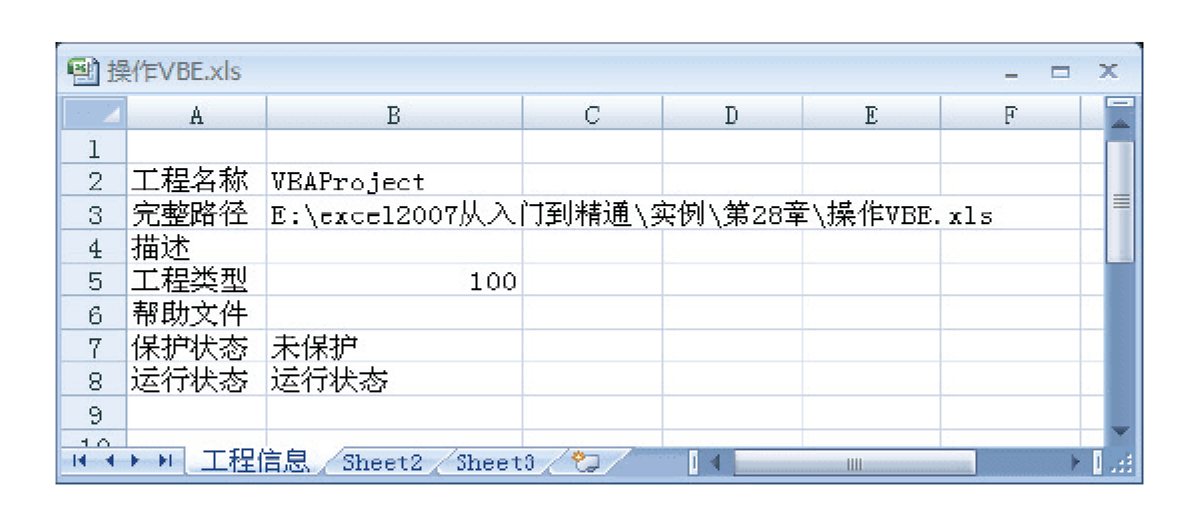


图 28-6　工程信息

执行以上代码时，如果 Excel 中的安全设置不允许访问工程对象模型，将产生出错误。在程序中使用 On Error 捕获出错信息，并弹出提示框提示用户。

28.3.2　查看部件

在 VBE 环境中，左侧显示的【工程】子窗口列出了工程中的所有部件，包括 Excel 对象、窗体、模块和类模块。

在 VBA 代码中，VBComponent 对象代表一个包含在工程中的组件，可以使用 Type 属性找出 VBComponent 对象所引用的部件类型，其返回值为以下常量。

- vbext_ct_ClassModule：类模块；
- vbext_ct_MSForm Microsoft：窗体；
- vbext_ct_StdModule：标准模块；
- vbext_ct_Document：Excel 对象（工作表、图表、Thisworkbook 对象）。

例如，以下代码可显示当前工程的全部部件，以及部件的类型。

```
Sub 查看部件()
    Dim oVBP As VBIDE.VBProject, oVBC As VBComponent
    Dim r As Long, str1 As String
    On Error Resume Next
    Set oVBP = ActiveWorkbook.VBProject
    If Err <> 0 Then
        MsgBox "当前安全设置不允许运行本过程，请修改安全设置！", _
            vbCritical + vbOKOnly, "警告"
        On Error GoTo 0
        Exit Sub
    End If
    With Worksheets("工程部件")
        .Columns("1:2").Clear
        .Range("A1:B1") = Array("类型", "名称")
        .Range("A1:B1").Font.Bold = True
        r = 1
        For Each oVBC In oVBP.VBComponents   '循环处理每个部件
            r = r + 1
            Select Case oVBC.Type            '部件类型
                Case vbext_ct_Document
                    str1 = "Excel 对象"
                Case vbext_ct_MSForm
                    str1 = "窗体"
                Case vbext_ct_StdModule
                    str1 = "模块"
                Case vbext_ct_ClassModule
                    str1 = "类模块"
            End Select
            .Cells(r, 1) = str1
            .Cells(r, 2) = oVBC.Name         '部件名称
        Next
        .Columns("A:B").AutoFit
```

```
    End With
End Sub
```

执行以上代码，将在工作表“工程部件”中看到当前工程的所有部件及其类型，如图 28-7 所示。

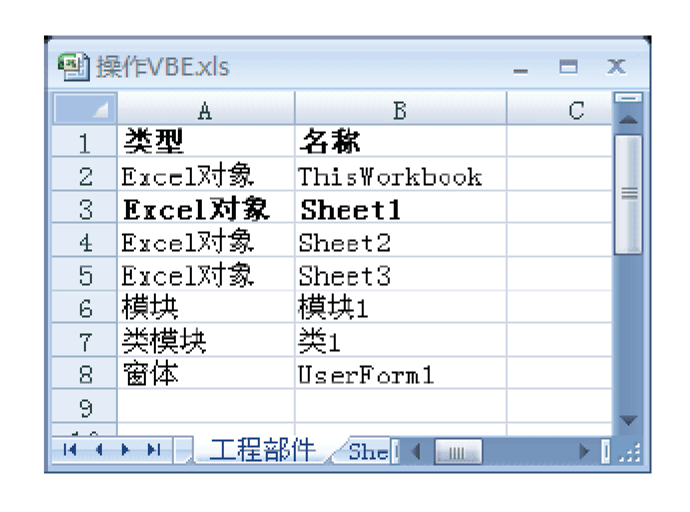


图 28-7　查看部件

28.3.3　查看引用

在 Excel 应用程序中，经常需要引用多个外部类型库（如引用 VBA、Word 等类型库）。在 VBE 环境中，通过单击主菜单【工具】|【引用】命令，打开【引用】对话框可查看当前工程的引用。

在 VBE 对象模型中，用 Reference 对象表示类型库或工程的引用。所有引用在 References 集合对象中。通过 Reference 对象的相关属性可获取或检查工程中的引用。

例如以下代码：

```
Sub 查看引用()
    Dim oVBP As VBIDE.VBProject, oRef As Reference, r As Long

    On Error Resume Next
    Set oVBP = ActiveWorkbook.VBProject
    If Err <> 0 Then
        MsgBox "当前安全设置不允许运行本过程，请修改安全设置！", _
            vbCritical + vbOKOnly, "警告"
        On Error GoTo 0
        Exit Sub
    End If

    With ActiveSheet
        .Columns("A:B").Clear                 '清除 A、B 两列的数据
        .Range("A1:B1") = Array("引用名称", "描述", "文件位置")     '填充表头
        .Range("A1:B1").Font.Bold = True
        r = 1
        For Each oRef In oVBP.References     '循环处理每个引用
            r = r + 1
            .Cells(r, 1) = oRef.Name          '引用名称
            .Cells(r, 2) = oRef.Description   '描述
            .Cells(r, 3) = oRef.FullPath      '保存位置
```

```
        Next
        .Columns("A:C").AutoFit
    End With
End Sub
```

执行以上代码，将在工作表“引用”中列出当前工程的所有引用，如图 28-8 所示。

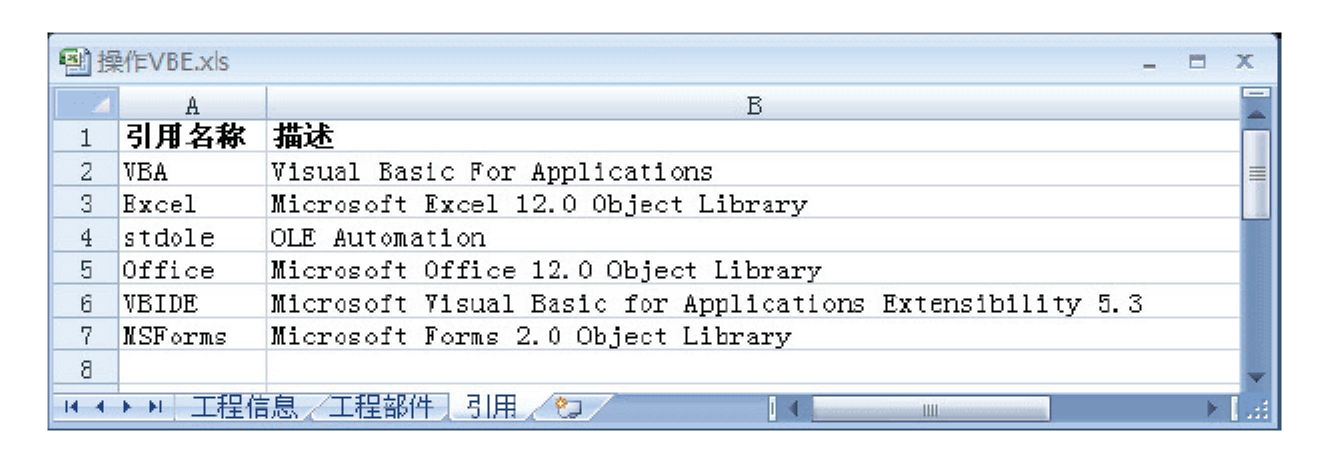

	A	B
1	引用名称	描述
2	VBA	Visual Basic For Applications
3	Excel	Microsoft Excel 12.0 Object Library
4	stdole	OLE Automation
5	Office	Microsoft Office 12.0 Object Library
6	VBIDE	Microsoft Visual Basic for Applications Extensibility 5.3
7	MSForms	Microsoft Forms 2.0 Object Library
8		

图 28-8　查看引用

28.4　用 VBA 控制 VBA 代码

CodeModule 对象在诸如窗体，类或文档等部件之后表示程序代码。编写 VBA 代码，通过 CodeModule 对象的属性和方法可控制工程中的 VBA 代码。

28.4.1　查看 VBA 过程名

在工程中，Excel 工作表、用户窗体、模块和类模块中都可以编写 VBA 代码。要显示工程中所有的过程名，需要循环处理几种部件。通过每个部件的 CodeModule 属性，获取该部件的 VBA 代码，再使用 CodeModule 对象的 ProcOfLine 方法获取指定行的过程名称。具体的代码如下：

```
Sub VBA过程名()
    Dim oVBP As VBIDE.VBProject, oVBC As VBComponent
    Dim oCM As CodeModule, strProcName As String
    Dim r As Long, r1 As Long, r2 As Long

    On Error Resume Next
    Set oVBP = ActiveWorkbook.VBProject
    If Err <> 0 Then
        MsgBox "当前安全设置不允许运行本过程，请修改安全设置！", _
            vbCritical + vbOKOnly, "警告"
        On Error GoTo 0
        Exit Sub
    End If
    With Worksheets("VBA过程名")
        .Columns("A:B").Clear                         '清除A、B两列的数据
        .Range("A1:B1") = Array("部件名称", "过程名称")    '填充表头
        .Range("A1:B1").Font.Bold = True
```

```
        r = 1
        For Each oVBC In oVBP.VBComponents               '循环处理每个部件
            Set oCM = oVBC.CodeModule                    '获取组件的代码模块对象
            r1 = oCM.CountOfDeclarationLines + 1         '跳过声明部分的代码
            r2 = oCM.CountOfLines                        '获取总的代码行数
            Do While r1 < r2
                r = r + 1
                .Cells(r, 1) = oVBC.Name
                .Cells(r, 2) = oCM.ProcOfLine(r1, vbext_pk_Proc)
                                                         '当前行的过程名
                r1 = r1 + oCM.ProcCountLines(oCM.ProcOfLine _
                    (r1, vbext_pk_Proc), vbext_pk_Proc) '下一过程
            Loop
        Next
        .Columns("A:B").AutoFit
    End With
End Sub
```

执行以上代码，将在工作表“VBA 过程名”中显示各部件中编写的 VBA 过程名，如图 28-9 所示。

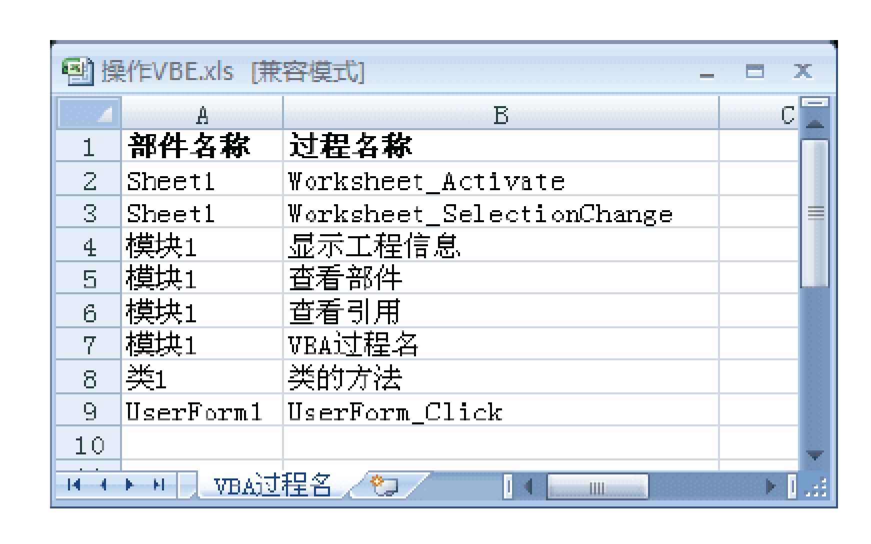


图 28-9　VBA 过程名

28.4.2　查看 VBA 代码

在 28.4.1 节的代码中，使用 CodeModule 对象的 ProcOfLine 方法取得指定行的过程名称。使用 CodeModule 对象的 Lines 属性可获取指定行区域的代码。Lines 属性的语法格式如下：

```
object.Lines(startline, count) As String
```

该属性将返回代码模块中从 startline 行号开始的 count 行代码。

例如，以下代码可将当前工程中各部件中的代码按过程分别显示在工作表中。

```
Sub 查看 VBA 代码()
    Dim oVBP As VBIDE.VBProject, oVBC As VBComponent
    Dim oCM As CodeModule, strProcName As String
    Dim r As Long, r1 As Long, r2 As Long, r3 As Long

    On Error Resume Next
```

```
    Set oVBP = ActiveWorkbook.VBProject
    If Err <> 0 Then
        MsgBox "当前安全设置不允许运行本过程，请修改安全设置！", _
            vbCritical + vbOKOnly, "警告"
        On Error GoTo 0
        Exit Sub
    End If

    With Worksheets("VBA 代码")
        .Columns("A:C").Clear                    '清除 A、B 两列的数据

        .Range("A1:C1") = Array("部件名称", "过程名称", "具体代码") '填充表头
        .Range("A1:C1").Font.Bold = True
        r = 1
        For Each oVBC In oVBP.VBComponents       '循环处理每个组件
            Set oCM = oVBC.CodeModule
            r1 = oCM.CountOfDeclarationLines + 1 '跳过声明部分的代码
            r2 = oCM.CountOfLines                '获取总的代码行数
            Do While r1 < r2
                r = r + 1
                .Cells(r, 1) = oVBC.Name
                .Cells(r, 2) = oCM.ProcOfLine(r1, vbext_pk_Proc)
                                                 '当前行的过程名
                 r3 = oCM.ProcCountLines(oCM.ProcOfLine(r1, vbext_pk_Proc),
_
                   vbext_pk_Proc)                '下一过程
                .Cells(r, 3) = oCM.Lines(r1, r3) '获取指定区域的代码
                r1 = r1 + r3
            Loop
        Next
        .Columns("A:C").AutoFit
    End With
End Sub
```

执行以上代码，将在工作表“VBA 代码”中分别中显示部件名称、过程名称和每个过程的 VBA 代码，如图 28-10 所示。

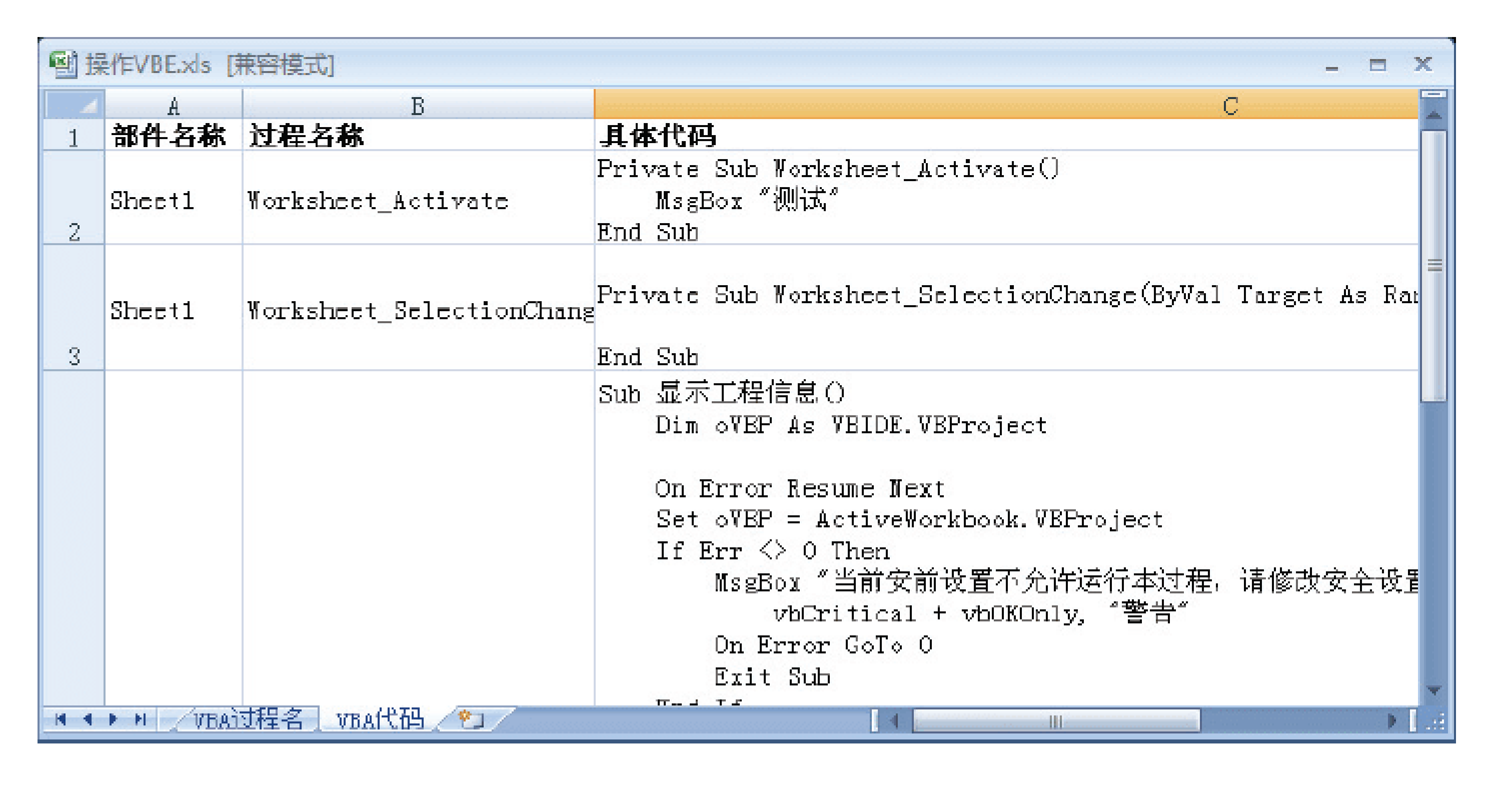


图 28-10　显示 VBA 代码

28.4.3　导出代码

使用 VBComponent 对象的 Export 方法，可将组件导出为单个文件。其语法格式如下：

```
object.Export (filename)
```

参数 filename 用来指定部件输出为文件的文件名。

当使用 Export 方法将组件存为文件时，使用的文件名不能重名；否则，将产生错误。

以下代码将导出用户指定部件的 VBA 代码。在导出代码时，由 VBA 代码根据部件的类型自动添加扩展名。

```
Sub 导出代码()
    Dim strName As String, str1 As String
    Dim oVBP As VBIDE.VBProject, oVBC As VBComponents

    On Error Resume Next
    Set oVBP = ActiveWorkbook.VBProject
    If Err <> 0 Then
        MsgBox "当前安全设置不允许运行本过程，请修改安全设置！", _
            vbCritical + vbOKOnly, "警告"
        On Error GoTo 0
        Exit Sub
    End If

    Set oVBC = oVBP.VBComponents
    strName = Application.InputBox(prompt:="输入需要导出代码的部件名称：", _
        Title:="输入名称", Type:=2)
    If strName = "False" Or strName = "" Then Exit Sub

    With oVBC(strName)
        Select Case .Type                               '根据部件类型生成扩展名
        Case vbext_ct_MSForm
            str1 = ".frm"
        Case vbext_ct_StdModule
            str1 = ".bas"
        Case vbext_ct_ClassModule
            str1 = ".cls"
        Case Else
            MsgBox "不能导出该部件的代码！", vbCritical + vbOKOnly, "警告"
            Exit Sub
        End Select
        strName = strName & str1                        '导出文件名
        .Export ThisWorkbook.Path & "\" & strName       '导出文件
    End With
    Set oVBC = Nothing
    Set oVBP = Nothing
End Sub
```

执行以上代码，将弹出如图 28-11 所示的对话框。在该对话框中输入要导出部件的名称，单击【确定】按钮，即可在工作簿所在的文件夹生成对应的 VBA 代码文件。

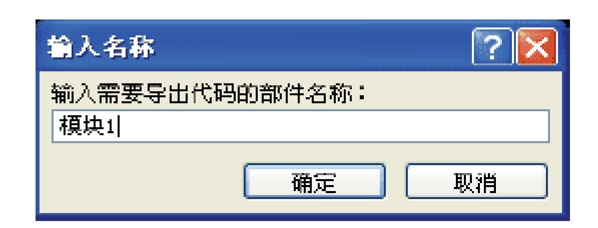


图 28-11　输入导出部件的名称

28.4.4　导入代码

与导出代码对应，可使用 VBComponents 集合对象的 Import 方法，将单个文件添加到工程中，作为窗体、模块或类模块。其语法格式如下：

```
object.Import(filename)
```

参数 filename 表示导入文件的路径及文件名。

提示： 如果工程中有相同名称的部件存在，导入操作将产生错误。

使用以下代码，可导入用户选择的文件：

```
Sub 导入代码()
    Dim strName As String, strFName As String
    Dim oVBP As VBIDE.VBProject, oVBC As VBComponents

    On Error Resume Next
    Set oVBP = ActiveWorkbook.VBProject
    If Err <> 0 Then
        MsgBox "当前安全设置不允许运行本过程，请修改安全设置！", _
            vbCritical + vbOKOnly, "警告"
        On Error GoTo 0
        Exit Sub
    End If

    Set oVBC = oVBP.VBComponents
    strFName = Application.GetOpenFilename( _
        filefilter:="VB 文件(*.bas;*.cls;*.frm),*.bas;*.cls;*.frm", _
        Title:="导入")              '获取导入文件名

    If strFName = "False" Or strFName = "" Then Exit Sub

    On Error Resume Next
    oVBC.Import strFName            '导入选定文件
    If Err <> 0 Then                '显示错误信息
        MsgBox Err.Number & ":" & Err.Description
    End If
    Set oVBC = Nothing
    Set oVBP = Nothing
End Sub
```

执行以上代码，将弹出如图 28-12 所示的对话框，选择要导入的 VB 文件，单击【打开】按钮即可将其导入到当前工程中。

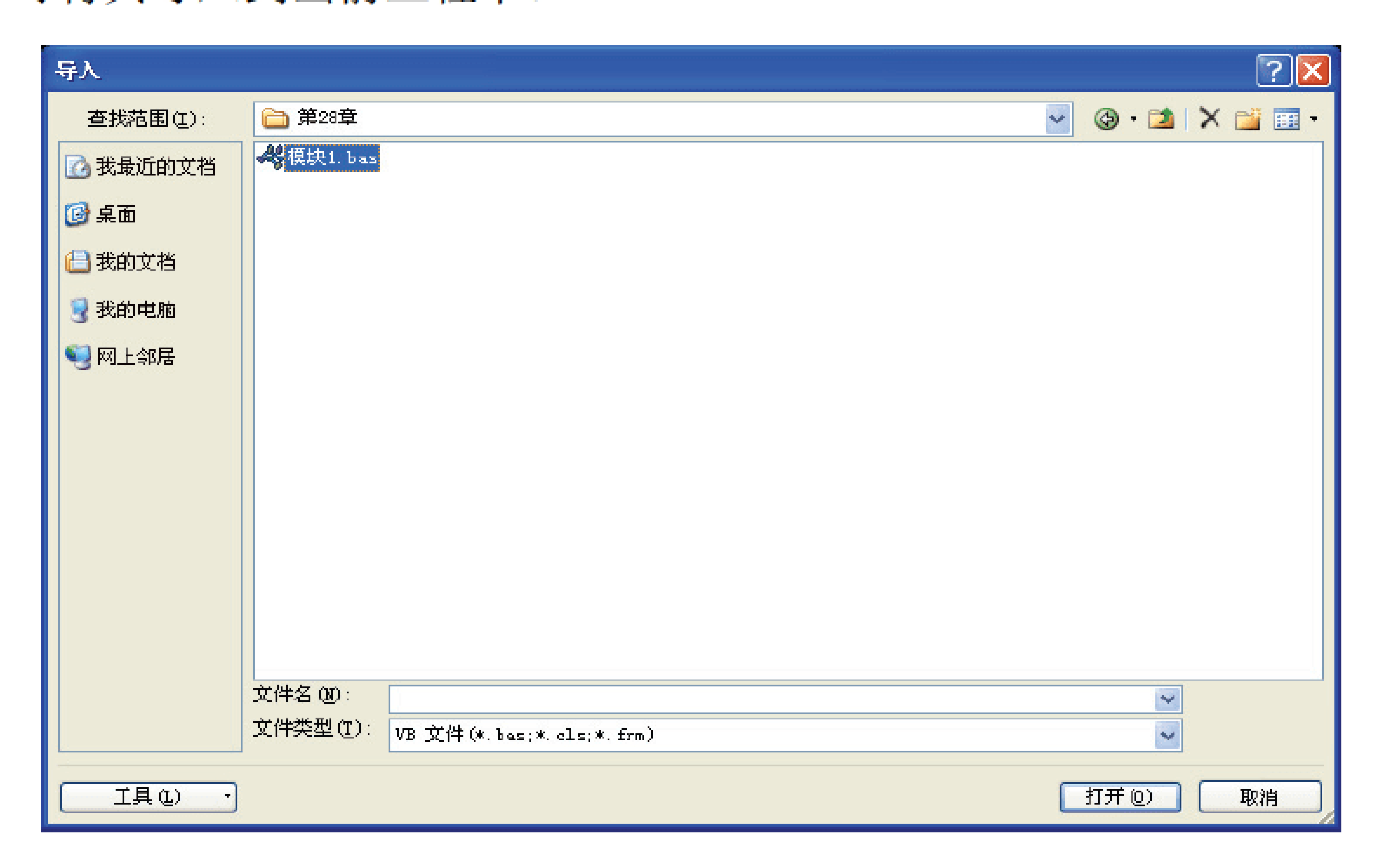


图 28-12　导入代码

28.4.5　在代码中搜索

当 Excel 应用程序的代码很多时，要查找代码中的某个关键字可单击主菜单中的【编辑】|【查找】命令，打开如图 28-13 所示的【查找】对话框，在其中输入查找内容，单击【查找下一个】按钮，就可在设置的范围内进行查找。

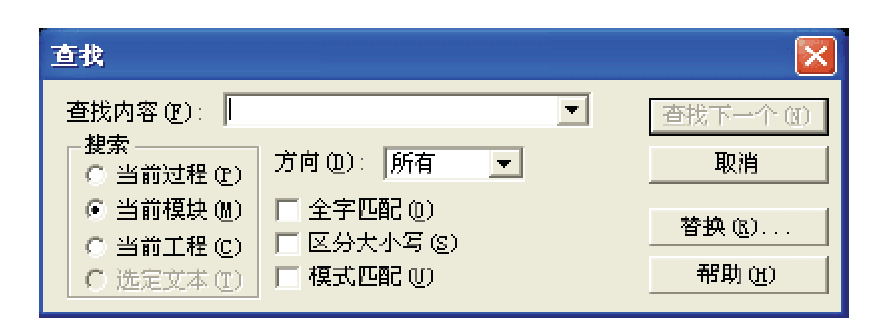


图 28-13　【查找】对话框

如果需要将具有该关键字的代码集中在一起进行比较，使用【查找】对话框就不太方便了。这时，可编写 VBA 代码，使用 CodeModule 对象的 Lines 属性对 VBA 代码进行逐行查找，并将找到包含关键字的代码复制到 Excel 工作表中。

通过 Lines 属性可获得一个字符串，可使用 InStr 函数在该字符串中查找是否存在子串。编写以下代码，可完成代码的搜索功能：

```
Sub 代码搜索()
    Dim oVBP As VBIDE.VBProject, oVBC As VBComponent, oCM As CodeModule
    Dim strProc As String, r As Long, i As Long, str1 As String

    On Error Resume Next
```

```
    Set oVBP = ActiveWorkbook.VBProject
    If Err <> 0 Then
        MsgBox "当前安全设置不允许运行本过程，请修改安全设置！", _
            vbCritical + vbOKOnly, "警告"
        On Error GoTo 0
        Exit Sub
    End If

    strProc = Application.InputBox(prompt:="请输入需要查找的关键字：", _
        Title:="代码搜索", Type:=2)
    If strProc = "False" Or strProc = "" Then Exit Sub

   With Worksheets("代码搜索结果")
        .Cells.Clear
        .Range("A1:B1") = Array("过程名", "代码") '填充表头
        .Range("A1:B1").Font.Bold = True
        r = 2
        For Each oVBC In oVBP.VBComponents
            Set oCM = oVBC.CodeModule               '处理部件代码模块
            For i = 1 To oCM.CountOfLines           '循环处理每一行代码
                str1 = LTrim(oCM.Lines(i, 1))
                If InStr(UCase(str1), UCase(strProc)) > 0 Then
                    .Cells(r, 1) = oCM.ProcOfLine(i, vbext_pk_Proc)   '过程名
                    .Cells(r, 2) = str1             '代码
                    r = r + 1
                End If
            Next
        Next
    End With
End Sub
```

执行以上代码，将弹出如图 28-14 所示的对话框，在该对话框中输入搜索的关键字（如 CodeModule）。

单击【确定】按钮，程序将获取各部件的代码模块，对每一行代码进行查找，找到相应的关键字后，再查找所在行的过程名，最后将相关信息显示在工作表中。如图 28-15 所示。

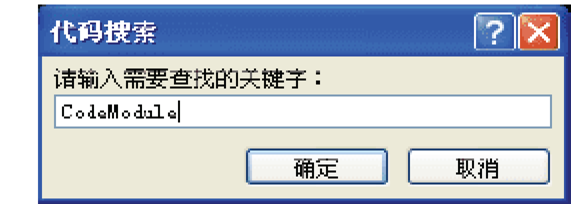
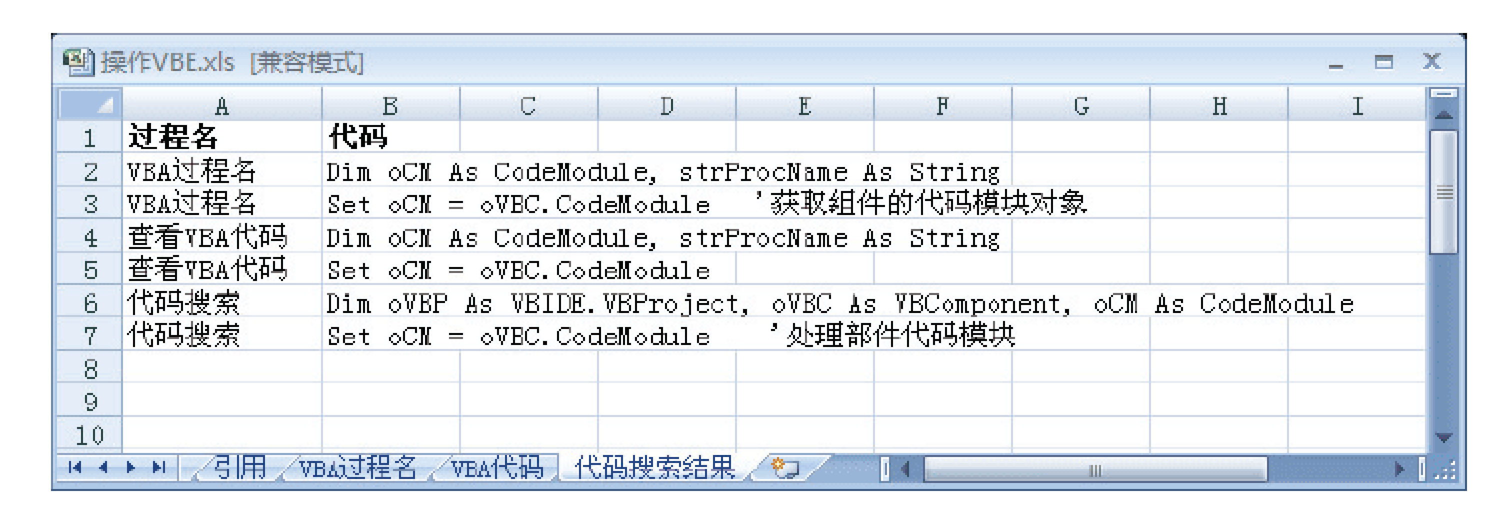


图 28-14 【代码搜索】对话框



	A	B
1	过程名	代码
2	VBA过程名	Dim oCM As CodeModule, strProcName As String
3	VBA过程名	Set oCM = oVBC.CodeModule '获取组件的代码模块对象
4	查看VBA代码	Dim oCM As CodeModule, strProcName As String
5	查看VBA代码	Set oCM = oVBC.CodeModule
6	代码搜索	Dim oVBP As VBIDE.VBProject, oVBC As VBComponent, oCM As CodeModule
7	代码搜索	Set oCM = oVBC.CodeModule '处理部件代码模块

图 28-15 搜索结果

28.5　动态添加 VBA 代码

通过调用 VBE 对象模型中的相应对象，可动态地创建模块、类模块、用户窗体及用户窗体中的控件，并可以动态地创建 VBA 代码。

28.5.1　增加模块

使用 VBComponents 集合对象的 Add 方法，可将一个对象添加到集合。该方法的语法格式如下：

```
object.Add(component)
```

参数 component 用来表示要添加的组件类型。对于 VBComponents 集合，则为表示类模块、窗体、标准模块的列举常数，具体常数如下所述。

- vbext_ct_ClassModule：将一个类模块添加到集合。
- vbext_ct_MSForm：将窗体添加到集合。
- vbext_ct_StdModule：将标准模块添加到集合。

使用以下代码可向工程中添加一个模块，模块名由用户输入。

```
Sub 增加模块()
    Dim oVBP As VBIDE.VBProject, oVBC As VBComponent
    Dim strMName As String

    On Error Resume Next
    Set oVBP = ActiveWorkbook.VBProject
    If Err <> 0 Then
        MsgBox "当前安全设置不允许运行本过程，请修改安全设置！", _
            vbCritical + vbOKOnly, "警告"
        On Error GoTo 0
        Exit Sub
    End If

    strMName = Application.InputBox(prompt:="请输入新增加模块的名称：", _
        Title:="模块名称", Type:=2)
    If strMName = "False" Or strMName = "" Then Exit Sub

    For Each oVBC In oVBP.VBComponents                 '检查是否已存在同名的模块
        If oVBC.Name = strMName Then
            MsgBox ("名称"" & strMName & ""已经存！请重新输入新的名称！"), _
                vbCritical + vbOKOnly, "警告"
            Exit Sub
        End If
    Next
    With oVBP.VBComponents.Add(vbext_ct_StdModule)  '增加模块
        .Name = strMName                               '设置模块名称
```

```
    End With
    MsgBox "增加模块成功！", vbInformation + vbOKOnly, "提示"
End Sub
```

执行以上代码，将弹出如图 28-16 所示对话框。在该对话框中输入模块的名称，单击【确定】按钮，即可向工程中添加一个模块。在【工程资源管理器】窗口中可看到新增的模块，如图 28-17 所示，该模块为一个空模块，还没有 VBA 代码。

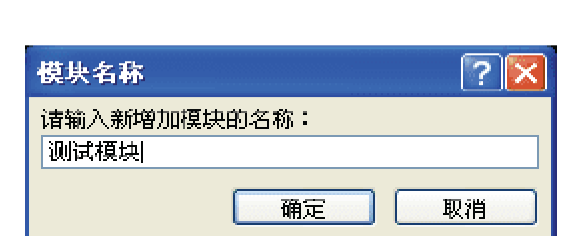


图 28-16　输入模块名

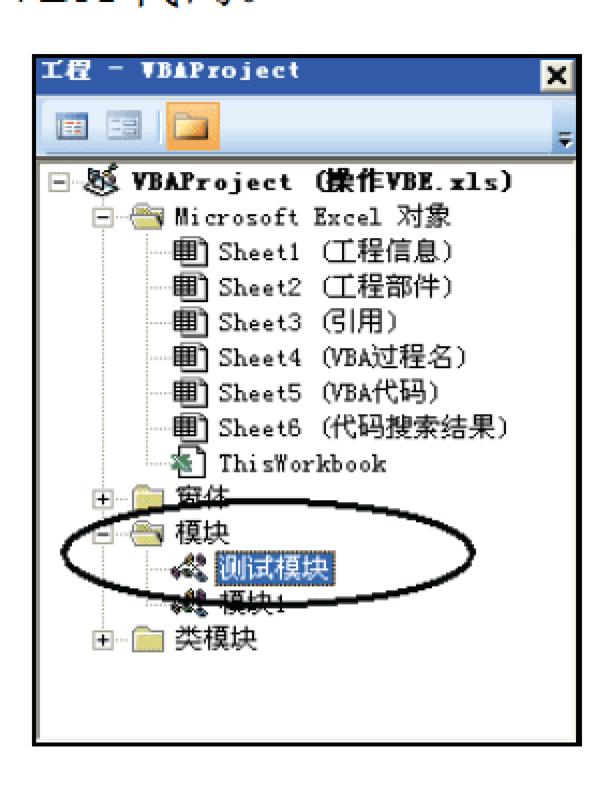


图 28-17　新增的模块

提示：如果要增加类模块，只需修改 Add 方法中的参数为以下形式即可：

```
VBComponents.Add(vbext_ct_ClassModule)
```

28.5.2　向模块中添加代码

使用 InsertLines 方法可在一个代码块的某个指定位置，插入一行或多行的代码。其语法格式如下：

```
object.InsertLines(line, code)
```

以下代码在用户指定的模块中插入一个过程代码：

```
Sub 插入代码()
    Dim oVBP As VBIDE.VBProject, oVBC As VBComponent
    Dim strMName As String, str1 As String

    On Error Resume Next
    Set oVBP = ActiveWorkbook.VBProject
    If Err <> 0 Then
        MsgBox "当前安全设置不允许运行本过程，请修改安全设置！", _
            vbCritical + vbOKOnly, "警告"
        Exit Sub
    End If

    strMName = Application.InputBox(prompt:="请输入插入代码的模块名称：", _
```

```
        Title:="模块名称", Type:=2)
    If strMName = "False" Or strMName = "" Then Exit Sub

    Set oVBC = oVBP.VBComponents(strMName)
    If Err <> 0 Then
        MsgBox "输入的模块名称错误！", vbCritical + vbOKOnly, "警告"
        Exit Sub
    End If

    str1 = "Sub 自动添加过程()" & vbNewLine & _
         Space(4) & "Dim str1 As String" & vbNewLine & _
         Space(4) & "str1=""" & "由 VBA 代码动态添加的测试代码！" & """" & vbNewLine
         & _
         Space(4) & "MsgBox str1" & vbNewLine & _
         "End Sub"

    oVBC.CodeModule.InsertLines 1, str1
    Set oVBC = Nothing
    Set oVBP = Nothing
End Sub
```

以上程序将需要插入到模块中的代码保存到字符串 str1 中，再调用 InsertLines 方法将其插入到指定模块。

执行以上代码，首先将弹出如图 28-18 所示对话框，在该对话框中输入需要插入代码的模块名称，单击【确定】按钮即可在该模块中插入代码，如图 28-19 所示。

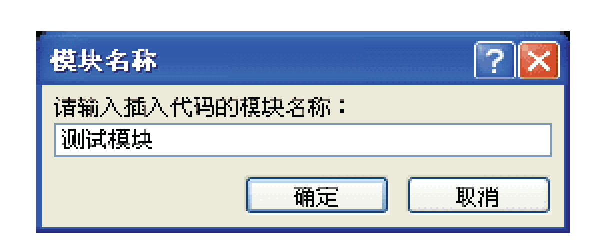


图 28-18　输入模块名

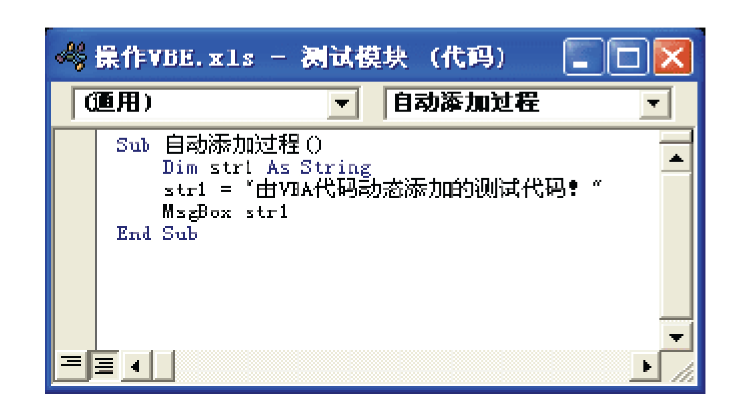


图 28-19　自动插入的代码

28.5.3　工作表中动态增加按钮

在 Excel 操作环境中，【开发工具】选项卡的【控件】组中提供了向工作表中插入的控件。最常用的是向工作表中添加一个按钮，并为按钮指定一个宏，让用户可以方便地在 Excel 界面中执行 VBA 代码。

通过 VBE 对象模型，可以让用户单击工作表中的按钮来动态添加按钮。要使用 VBA 代码动态向工作表中添加按钮，可使用 OLEObject 对象的相关方法。

OLEObject 对象代表工作表上的一个 ActiveX 控件或链接、嵌入的 OLE 对象。OLEObject 对象是 OLEObjects 集合的成员。OLEObjects 集合在一张工作表上包含所有的 OLE 对象。

使用 OLEObjects 集合的 Add 方法可向工作表中添加新的 OLE 对象。Add 方法的语法

格式如下：

```
表 达 式 .Add(ClassType, FileName, Link, DisplayAsIcon, IconFileName,
IconIndex, IconLabel, Left, Top, Width, Height)
```

其中参数 ClassType 为要创建对象的程序标识符。其余参数都可省略。

通过手动方式向工作表中添加一个 ActiveX 按钮，在编辑栏中可看到如下内容：

```
=EMBED("Forms.CommandButton.1","")
```

其中的“Forms.CommandButton.1”就是按钮的 ClassType。

了解以上知识后，就可编写工作表中动态添加按钮的代码。具体代码如下：

```
Sub 动态添加按钮()
    Dim oVBP As VBIDE.VBProject, ws1 As Worksheet, i As Integer
    Dim oBtn As Object, obj As OLEObject, sCode As String, r As Long

    On Error Resume Next
    Set oVBP = ActiveWorkbook.VBProject
    If Err <> 0 Then
        MsgBox "当前安全设置不允许运行本过程，请修改安全设置！", _
            vbCritical + vbOKOnly, "警告"
        Exit Sub
    End If

    i = 1
    Set ws1 = Worksheets("动态按钮")
    For Each obj In ws1.OLEObjects        '判断已添加了几个 ActiveX 按钮
        If obj.Name = "cmd" & i Then
            i = i + 1
        End If
    Next

    Set oBtn = ws1.OLEObjects.Add("Forms.CommandButton.1")
                                          '添加一个 ActiveX 按钮

    With oBtn                             '设置按钮的属性
        .Left = 110 * (i - 1) + 110       '动态设置按钮左侧坐标
        .Top = 40
        .Width = 100
        .Height = 30
        .Object.Caption = "动态按钮" & I  '按钮显示名称
        .Name = "cmd" & I                 '按钮代码名称
    End With
                                          '生成按钮的单击事件代码
    sCode = "Sub cmd" & i & "_Click()" & vbNewLine & _
        "   MsgBox ""你单击了第" & i & "个动态按钮！"" " & vbNewLine & _
        "End Sub"
                                          '添加代码到工作表代码部分
    With oVBP.VBComponents(ws1.CodeName).CodeModule
        r = .CountOfLines + 1             '代码模块总行数加 1
```

```
        .InsertLines 1, sCode          '在最后一行处插入新的代码
    End With
End Sub
```

以上代码首先判断当前工作表中已有几个 ActiveX 按钮，取得序列号后，方便给后续添加的按钮命名（如果动态按钮具有相同的名称，则动态按钮的事件过程名将相同，从而导致程序出错）。接着使用 OLEObjects 集合的 Add 方法添加一个 ActiveX 按钮，并设置按钮的位置、名称，然后向工作表代码中添加该按钮的事件过程代码。

在工作表中添加一个按钮，设置名称为“动态按钮”，为该按钮指定宏为“动态按钮”（前面编写的代码）。单击 2 次该按钮，将在工作表中添加 2 个按钮，如图 28-20 所示。单击工作表中的【动态按钮 2】，将执行该按钮的单击事件代码，弹出如图 28-21 所示的提示对话框。



图 28-20　动态添加按钮



图 28-21　提示

在 VBE 中双击工作表“动态按钮”，可看到由 VBA 代码添加的按钮事件代码，如图 28-22 所示。

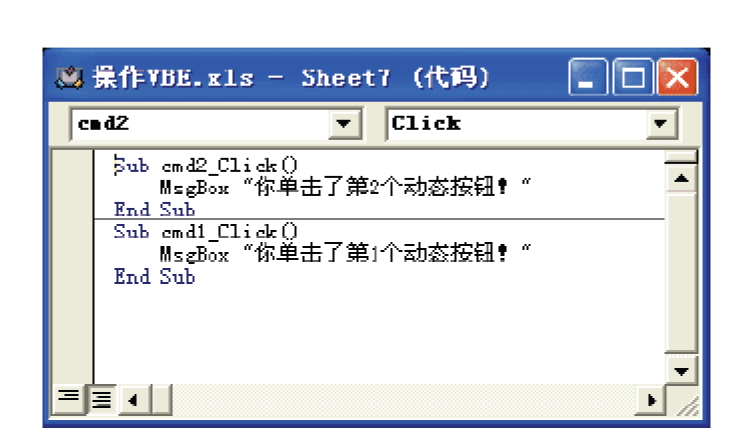


图 28-22　动态添加的代码

28.5.4　创建动态用户窗体

与添加模块类似，使用 VBComponents 集合的 Add 方法也可向工程中增加用户窗体，只需将 Add 方法的参数设置为 Vbext_ct_MSForm 即可。

使用 Add 方法添加的只是一个空白用户窗体，还需要向窗体中添加各种控件，窗体才能完成相应的功能。

使用 Designer 属性可返回窗体设计器，通过设计器访问 Controls 集合，向窗体中添加控件。例如，使用以下代码可向窗体中添加一个按钮。

```
Set oBtn = oWindow.Designer.Controls.Add("Forms.CommandButton.1")
```

如果要添加其他窗体控件，只需将 Add 参数中的“Forms.CommandButton.1”替换为

其他控件的类型标识符即可。

下面的代码将创建一个用户窗体，并向窗体中添加两个按钮。

```
“创建窗体”按钮的 VBA 代码如下：
Sub 创建窗体()
    Dim oVBP As VBIDE.VBProject, obj As VBComponent
    Dim oWindow As Object, oBtn As Object, oChk As Object
    Dim sCode As String, r As Long

    On Error Resume Next
    Set oVBP = ActiveWorkbook.VBProject
    If Err <> 0 Then
        MsgBox "当前安全设置不允许运行本过程，请修改安全设置！", _
            vbCritical + vbOKOnly, "警告"
        On Error GoTo 0
        Exit Sub
    End If

    For Each obj In oVBP.VBComponents   '判断是否已有同名用户窗体
        If obj.Name = "myFrm" Then
            MsgBox "该动态窗体已经创建！" & vbNewLine _
                & "下面将删除该窗体，然后重新创建窗体！"
            oVBP.VBComponents.Remove obj
        End If
    Next

    Set oWindow = oVBP.VBComponents.Add(vbext_ct_MSForm)'添加一个用户窗体
    With oWindow                        '设置窗体的属性值
        .Properties("Caption") = "动态窗口"
        .Properties("Name") = "myFrm"
        .Properties("Width") = 300
        .Properties("Height") = 100
    End With

    Set oBtn = oWindow.Designer.Controls.Add("Forms.CommandButton.1")
                                        '添加一个按钮
    With oBtn                           '设置按钮的属性值
        .Name = "cmdOK"
        .Caption = "确定"
        .Left = 100
        .Top = 40
        .Width = 50
        .Height = 20
    End With
                                        '生成"确定"按钮的单击事件代码
    sCode = "Sub cmdOK_Click()" & vbNewLine & _
        "  msgbox ""你单击了【确定】按钮！""" & vbNewLine & _
        "End Sub" & vbNewLine

    Set oBtn = oWindow.Designer.Controls.Add("Forms.CommandButton.1")
```

```
    With oBtn
        .Name = "cmdClose"
        .Caption = "关闭"
        .Left = 160
        .Top = 40
        .Width = 50
        .Height = 20
    End With
         '生成【关闭】按钮的事件单击代码
    sCode = sCode & "Sub cmdClose_Click()" & vbNewLine & _
        "    Unload Me" & vbNewLine & _
        "End Sub"
         '将代码添加到窗体的代码中
    With oWindow.CodeModule
        r = .CountOfLines + 1
        .InsertLines r, sCode
    End With

    '将创建的窗体放入 UserForms 集合中，并显示出来
    VBA.UserForms.Add(oWindow.Name).Show
End Sub
```

以上代码首先判断用户窗体是否存在，接着使用 VBComponents 集合的 Add 方法创建一个用户窗体，再通过用户窗体的 Designer 对象添加两个按钮控件到窗体中，接着生成两个按钮的单击事件代码，并将代码添加到窗体中，最后将窗体添加到 UserForms 集合中，并显示出来。

在工作表中添加一个按钮，设置名称为“创建窗体”，为该按钮指定宏为“创建窗体”（前面编写的代码）。单击该按钮，将弹出如图 28-23 所示的用户窗体，单击【确定】按钮将弹出如图 28-24 所示的提示对话框。单击【关闭】按钮将关闭用户窗体。



图 28-23　生成的动态窗体



图 28-24　提示信息

第 29 章　使用 Windows API

Windows　API 是 Windows 应用程序接口的简称，是为开发者提供的开发 Windows 应用程序时的接口。Windows API 提供了功能众多的函数，可让程序员深入控制 Windows 操作系统的绝大部分内容。在 VBA 中也可调用 Windows　API 函数，扩展和优化基于 Office 的应用程序。

29.1　Windows API 基础

Windows API 是 Windows 操作系统提供的底层函数，使用这些函数可使开发者编写的代码进入操作系统核心。要在 Excel 中调用 API 函数，需要先了解 API 及其相关知识。

29.1.1　Windows API 概述

API 的英文全称是 Application Programming Interface，Windows API 也就是 Microsoft Windows 平台的应用程序编程接口。

使用这些 API 函数可以像搭积木一样编写 Windows 环境下的应用程序。但是，如果全部使用 API 函数进行应用程序开发，程序员就必须熟记一大堆常用的 API 函数，而且还得对 Windows 操作系统有深入的了解。这样，程序的开发效率将很低。

随着软件技术的不断发展，在 Windows 平台上出现了很多优秀的可视化编程环境，程序员可以采用“即见即所得”的编程方式来开发具有精美用户界面和功能强大的应用程序。这些优秀的可视化编程环境操作简单、界面友好（例如 VB、Visual C++、DELPHI 等），在这些工具中提供了大量的类库和各种控件，它们替代了 API 的神秘功能，事实上这些类库和控件都是构架在 Windows API 函数基础之上的，是封装了的 API 函数的集合。它们把常用的 API 函数组合在一起成为一个控件或类库，并赋予其方便的使用方法，所以极大地提高了 Windows 应用程序的开发效率。有了这些控件和类库，程序员便可以把主要精力放在程序整体功能的设计上，而不必过于关注技术细节。

如果要开发出更灵活、更实用、更具效率的应用程序，必然要涉及到直接使用 API 函数，虽然使用类库和控件会使应用程序的开发简单的多，但它们只提供 Windows 的一般功能，对于比较复杂和特殊的功能来说，使用类库和控件是非常难以实现的，这时就需要采用 API 函数来实现。

凡是在 Windows 工作环境下执行的应用程序，都可以调用 Windows API。当然，Office 应用程序同样也可调用 API 函数，以扩充应用程序的功能。

29.1.2　API 分类

每个 Windows 应用程序都直接或间接地调用 Windows API 提供的函数，通过这些函数可以保证所有在 Windows 中运行的应用程序都按照统一的方式运行。

Windows API 函数包含在一系列扩展名为 DLL 的动态链接库文件中，共有上千个 API 函数。对于数量众多的 API 函数，开发者不必刻意去研究每个 API 函数的用法，但是在需要的时候，至少应该知道它属于哪一类的 API 函数，这样才能正确查找和使用。

按照通常的划分标准，Windows API 函数分为下面 7 大类。

- ❑ 窗体管理类：这类 API 函数向应用程序提供了一些创建和管理用户界面的方法，可以使用它们来控制应用程序的界面。
- ❑ 窗体通用控制类：系统 SHELL 提供了一些控制，使用这些控制可以使窗体具有与众不同的外观，通用控制是由通用控制库 COMCTL32.DLL 提供的。
- ❑ SHELL 特性类：应用程序可以使用它们来增强系统 SHELL 各方面的功能。
- ❑ 图形设备接口（GDI32）：提供绘图、图形处理、使用显示设备等一系列的 API 函数。
- ❑ 系统服务类（Kernel32）：提供了访问操作系统提供的计算机资源的功能。这些函数包括控制内存、文件系统和系统上运行的资源的函数。
- ❑ 国际特性类：有助于编写国际化的应用程序，提供 Unicode 字符集和多语种支持。
- ❑ 网络服务类：允许网络上的不同计算机之间的不同应用程序之间进行通信，用于在各计算机上创建和管理共享资源的连接。

对于众多的 API 函数，开发人员不必刻意去研究每一个函数的用法，在需要的时候通过查看 API 帮助掌握其使用方法即可。

通过 Windows API 提供的函数，使用 VBA 可以利用 Windows 操作系统的强大功能，完成许多使用 VBA 语句难以完成的工作。本章将结合实例介绍 API 的使用方法，使读者掌握使用 API 的方法。

29.2　在 Excel 中使用 API

VBA 可以调用动态链接库（DLL）中的函数。在大多数情况下，这些 DLL 文件是由 C 或 C++之类的程序语言开发的。在 VBA 中调用这些函数时，需要进行一些特殊设置，使其能与 C 的数据类型进行数据交换。本节介绍具体的设置方法。

29.2.1　声明函数

由于大部分 DLL 及其文档最初是为 C/C++程序员编写的，所以调用 DLL 函数和调用 VBA 函数可能会有所不同。

1. 声明API

使用 VBA 的 Declare 语句，可在模块级别中声明对动态链接库（DLL）中外部过程的引用。Declare 语句的两种语法格式如下：

```
[Public | Private] Declare Sub name Lib "libname" [Alias "aliasname"]
[([arglist])]
[Public | Private] Declare Function name Lib "libname" [Alias "aliasname"]
[([arglist])] [As type]
```

Declare 语句的语法包含下面几部分的内容。

- Public：用于声明对所有模块中的所有其他过程都可以使用的过程。
- Private：用于声明只能在包含该声明的模块中使用的过程。
- Sub：表示该过程没有返回值。
- Function：表示该过程会返回一个可用于表达式的值。
- Name：任何合法的过程名。注意动态链接库的名称区分大小写。
- Lib：指明包含所声明过程的动态链接库或代码资源。所有声明都需要 Lib 子句。
- Libname：包含所声明的过程动态链接库名或代码资源名。
- Alias：表示将被调用的过程在动态链接库（DLL）中还有另外的名称。当外部过程名与某个关键字重名时，就可以使用这个参数。当动态链接库的过程与同一范围内的公用变量、常数或任何其他过程的名称相同时，也可以使用 Alias。如果该动态链接库过程中的某个字符不符合动态链接库的命名约定时，也可以使用 Alias。
- aliasname：动态链接库或代码资源中的过程名。如果首字符不是数字符号（#），则 aliasname 是动态链接库中该过程的入口处的名称。如果首字符是（#），则随后的字符必须指定该过程的入口处的顺序号。
- arglist：代表调用该过程时需要传递的参数的变量表。
- type：Function 过程返回值的数据类型；可以是 Byte、Boolean、Integer、Long、Currency、Single、Double、Date、String（只支持变长）或 Variant 的用户定义类型，或对象类型。

2. 参数传递

在 Declare 语句中，可通过 arglist 参数定义向 API 函数传递的参数名称、数据类型等。Arglist 参数的语法以及语法各个部分如下：

```
[Optional] [ByVal | ByRef] [ParamArray] varname[( )] [As type]
```

各部分的含义如下所述。

- Optional：表示参数不是必需的。如果使用该选项，则 arglist 中的后续参数都必需是可选的，而且必须都使用 Optional 关键字声明。如果使用了 ParamArray，则任何参数都不能使用 Optional。
- ByVal：表示该参数按值传递。
- ByRef：表示该参数按地址传递。ByRef 是 VBA 的默认选项。

- ParamArray：只用于 arglist 的最后一个参数，表示最后的参数是一个 Variant 元素的 Optional 的数组。使用 ParamArray 关键字可以提供任意数目的参数。ParamArray 关键字不能与 ByVal、ByRef 或 Optional 一起使用。
- varname：代表传给该过程的参数的变量名；遵循标准的变量命名约定。
- ( )：指明 varname 是一个数组。
- type：传递给该过程的参数的数据类型；可以是 Byte、Boolean、Integer、Long、Currency、Single、Double、Date、String（只支持变长）、Object、Variant 的用户自定义类型或对象类型。

默认情况下，VBA 中的参数通过引用来传递。因此，当 API 函数要求通过赋值来传递参数时，必须在该函数定义中包含关键字 ByVal。如果在函数定义中省略关键字 ByVal，有些情况下可能会导致无效页面的错误。在另外一些情况下可能会发生 VBA 运行时错误 49。

以引用方式传递参数会将该参数的内存地址传递给被调用的过程。如果被调用的过程改变参数的值，则将改变参数的唯一版本，因此当执行过程返回到主调过程时，该参数将含有更改后的值。

以赋值方式向 API 函数传递参数是传递该参数的一个副本；函数将使用该副本来代替参数进行操作。这样可以避免函数更改真实参数的内容。当执行进程返回到主调过程时，参数包含的值与调用其他过程之前相同。

由于以引用方式传递参数可以在内存中修改参数值，因此如果不正确地以引用方式传递参数，API 函数可能会改写它不应该写入的内存，从而导致错误或其他一些难以预料的结果。Windows 中有很多不应该被改写的值。例如，Windows 给每个窗体分配一个唯一的称为“句柄”的 32 位标识符。

注意：当所调用的外部过程需要一个值为 0 的字符串时，就要使用 vbNullString 常数。该常数与零长度字符串("")是不相同的。

3. 使用常量

有些 API 函数还需要定义它用到的常量及类型。应将常量、用户自定义类型的定义，与要使用它们的函数的 Declare 语句一起放在模块的声明部分。从 API 的相关文档可以找到各函数需要使用的常量和用户自定义类型。

有时可能需要向函数传递常量，以指明要求该函数返回何种信息。例如，GetSystemMetrics 函数接受 75 个常量之一，每个常量都指定操作系统的不同方面。函数返回什么信息取决于向它传递什么常量。为了能够调用 GetSystemMetrics 函数，在程序中不必包含所有 75 个常量，只包含要使用的那些常量即可。

29.2.2　使用 API 浏览器

从前面的介绍可知，在使用 API 函数之前，必须先声明 API 函数，另外，API 函数中的很多参数使用预定义的结构和常数。API 声明（包括结构、常数）必须放在窗体或模块

的“通用”（General）段中。

API 函数声明部分看起来就觉得很复杂，并且对字符需要区分大小写。API 函数的声明形式、结构和常数的定义可通过查 API 手册获得。但是手工输入很容易出错，一般情况下，可直接从 API 浏览器中复制过来，这样可以避免出现错误。下面以 VB6 开发环境中提供的 API 浏览器为例，介绍查询的方法。

（1）在 VB6 中单击主菜单【外接程序】|【API 浏览器】命令（或运行 APILOAD.EXE 程序，该程序可独立运行），打开如图 29-1 所示的窗体。

（2）在图 29-1 所示对话框中还未显示任何内容，需要先载入相应的 API 文件。单击菜单【文件】|【文本文件】命令，打开如图 29-2 所示对话框，选择 WIN32API.TXT，并单击【打开】按钮将该文件加载到 API 浏览器窗体中。

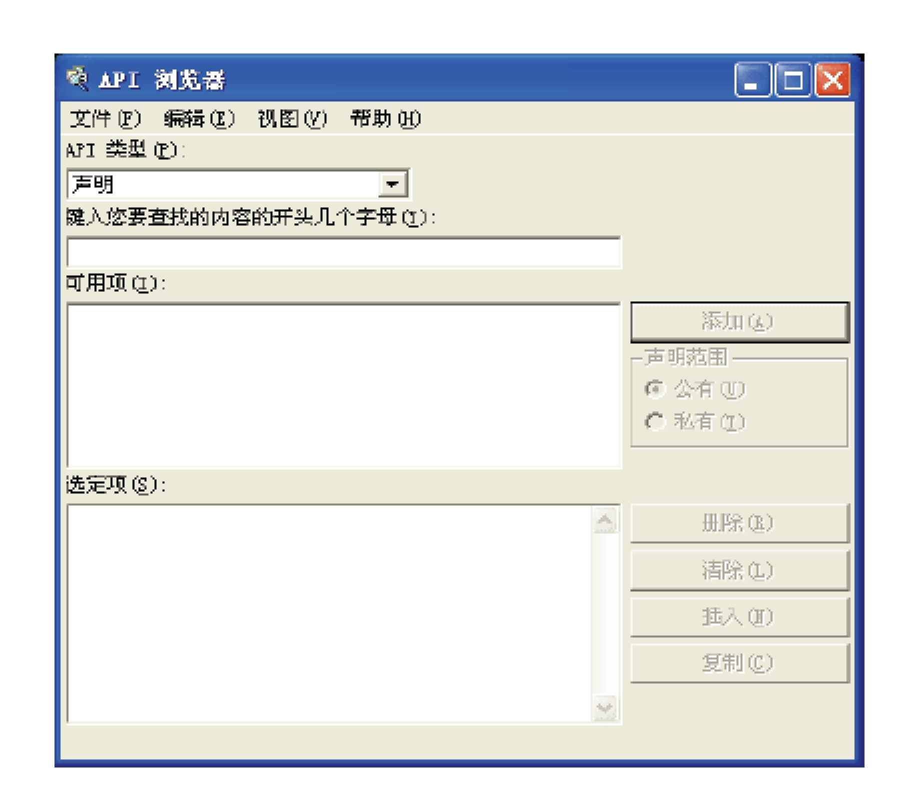


图 29-1 【API 浏览器】窗口

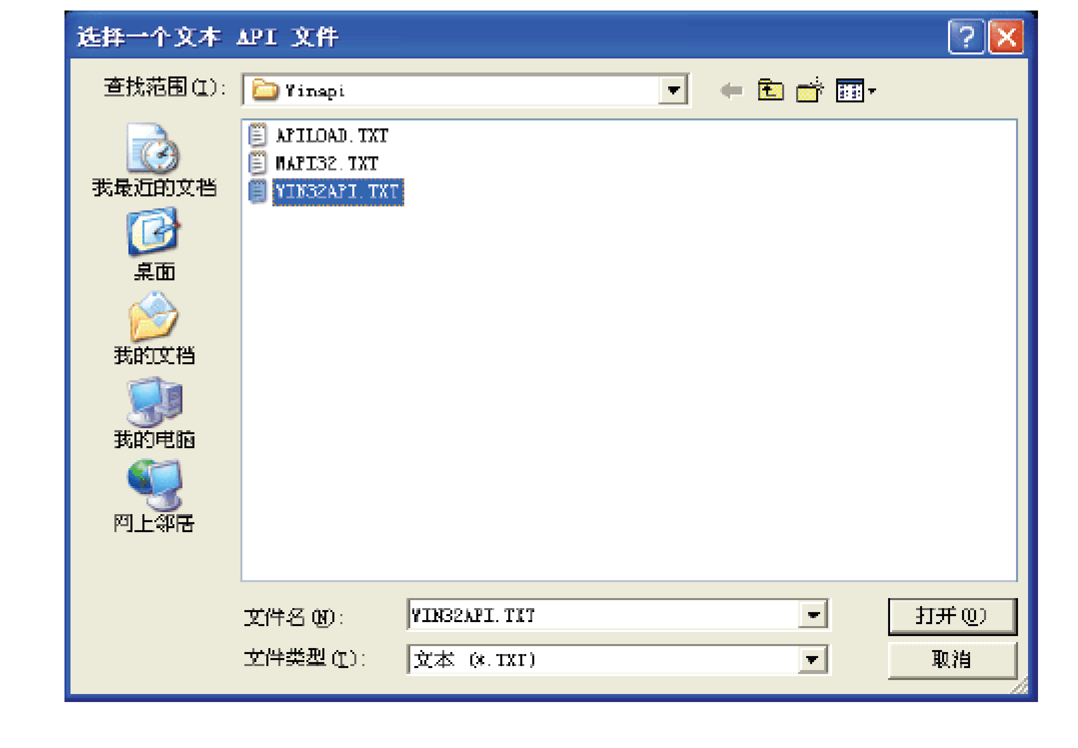


图 29-2 加载文本文件

（3）在【API 类型】下拉列表框中可以选择声明，然后在下面的文本框中输入部分字母 flash，在【可用项】列表框中将显示匹配的内容。

（4）在【可用项】列表框中双击 API 函数 FlashWindow，该项的详细内容将添加到下方的【选定项】列表框中，如图 29-3 所示。



图 29-3 查找 API 函数声明

（5）单击【复制】按钮，将【选定项】列表框中的内容复制到剪贴板中。

（6）切换到 Excel 的 VBE 环境中，将剪贴板中的内容粘贴到相应的模块中即可。

技巧：也可打开 WIN32API.TXT 文件，直接在该文件中查找相应的内容，找到后将其复制即可。

29.2.3　调用 API 函数

USER.DLL 动态链接库中包含一个 API 函数 FlashWindow，使用该函数可将指定的窗体闪烁，以提醒用户注意。下面以实例调用该函数来演示 API 函数的使用方法。本例需要使用以下两个 API 函数：

- FlashWindow 函数

该函数用来闪烁显示指定窗体。其声明格式如下：

```
Private Declare Function FlashWindow Lib "user32" ( _
   ByVal hWnd As Long, ByVal bInvert As Long) As Long
```

其中参数 hWnd 表示要闪烁显示的窗体的句柄，在 VBA 中，没有属性可以直接返回窗体的句柄，所以还需要使用 FindWindow 函数查询窗体的句柄。

- FindWindow 函数

该函数用来获取指定窗体的句柄。其声明格式如下：

```
Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" ( _
   ByVal lpClassName As Long, ByVal lpWindowName As String) As Long
```

该函数有两个参数，第一个是要找的窗体的类，第二个是要找的窗体的标题。在搜索的时候不一定两者都知道，但至少要知道其中的一个。有的窗体的标题是比较容易得到的，可以按标题进行搜索。但有的软件的标题不是固定的，可以按窗体类搜索。如果找到了满足条件的窗体，该函数将返回满足条件窗体的句柄，否则返回 0。

了解这两个 API 函数的作用、参数以后，就可以在 Excel VBA 中开始编写代码。具体操作步骤如下：

（1）在 Excel 中按快捷键 Alt+F11 进入到 VBE 环境中。

（2）单击菜单【插入】|【用户窗体】命令，向工程中增加一个用户窗体。

（3）双击用户窗体空白处，打开代码窗体。将 API 函数的 FlashWindow 的声明代码粘贴到窗体的声明部分。

```
Private Declare Function FlashWindow Lib "user32" ( _
   ByVal hwnd As Long, ByVal bInvert As Long) As Long
Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" ( _
   ByVal lpClassName As Long, ByVal lpWindowName As String) As Long
```

注意：在【API 浏览器】窗口中查到 API 函数的声明全部是使用 Public 声明其作用域，在用户窗体中，必须将其改为 Private。

（4）在用户窗体的声明部分再声明一个模块变量，用来保存窗体的句柄。

```
Dim hWnd As Long    '保存当前窗体的句柄
```

（5）在用户窗体的初始化事件（Initialize）中编写代码，获取窗体的句柄。具体代码如下：

```
Private Sub UserForm_Initialize()
    hWnd = FindWindow(0&, "UserForm1")
End Sub
```

使用 FindWindow 函数查找用户窗体的句柄时，因窗体标题比较容易获得，所以省略了窗体类的名称，直接传递一个长整型的 0 给函数。

（6）关闭代码窗体。在窗体中添加一个按钮，设置按钮的 Caption 属性为“闪烁窗体”。

（7）为按钮的单击事件编写代码如下：

```
Private Sub cmdFlash_Click()
    For i = 1 To 100
        Call FlashWindow(hWnd, True)
        DoEvents
    Next
    Call FlashWindow(hWnd, False)    '使窗体处于活动状态
End Sub
```

执行创建的用户窗体，如图 29-4 所示。单击【闪烁窗体】按钮，可看到用户窗体标题栏的颜色在不停闪烁。



图 29-4　闪烁窗体

29.3　制作特殊窗体

在 VBE 中，通过单击菜单【插入】|【用户窗体】命令向工程中插入的用户窗体是一个矩形。在有的应用程序中，为了增加程序的趣味性，可能需要制作非矩形的窗体。这时，只有通过 API 函数才能完成任务。

29.3.1　制作半透明窗体

有一些屏幕翻译软件为了不影响用户操作其他软件，通常将显示翻译结果的窗体设置

为半透明状。在 VBA 中要制作这种类型的窗体，需使用 API 函数。

要实现半透明窗体的效果，需要使用多个 API 函数，下面简单介绍这些 API 函数。

1. SetLayeredWindowAttributes函数

使用该函数设置窗体的透明效果。该函数的原型如下：

```
Private Declare Function SetLayeredWindowAttributes Lib "user32" _
    (ByVal hWnd As Long, ByVal crKey As Long, ByVal bAlpha As Byte, _
    ByVal dwFlags As Long) As Long
```

函数中各参数的含义如下面所述。

- hWnd：是透明窗体的句柄，通过 API 函数 FindWindow 取得指定窗体的句柄；
- crKey：为颜色值；
- bAlpha：是透明度，取值范围是 0～255；
- dwFlags：是透明方式，可以取以下两个值。
  - LWA_ALPHA：值为 2。设置为该值时，crKey 参数无效，bAlpha 参数有效。
  - LWA_COLORKEY：值为 1。设置为该值时，bAlpha 参数有效而窗体中的所有颜色为 crKey 的地方将变为透明。

提示：要使窗体拥有透明效果，首先要使用以下语句定义 WS_EX_LAYERED 常量：

```
WS_EX_LAYERED = 0x80000
```

2. GetWindowLong函数

用该函数能够获得指定窗体的信息，其函数原型如下：

```
Private Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" _
    (ByVal hWnd As Long, ByVal nIndex As Long) As Long
```

两个参数的含义分别如下面所述。

- hWnd：指定窗体的句柄；
- nIndex：需要获得的信息的类型。该参数可设置为以下值之一。
  - GWL_EXSTYLE：得到扩展的窗体风格；
  - GWL_STYLE：得到窗体风格；
  - GWL_WNDPROC：得到窗体回调函数的地址，或者句柄。得到后必须使用 CallWindowProc 函数来调用；
  - GWL_HINSTANCE：得到应用程序运行实例的句柄；
  - GWL_HWNDPARENT：得到父窗体的句柄；
  - GWL_ID：得到窗体的标识符；
  - GWL_USERDATA：得到和窗体相关联的 32 位的值（每一个窗体都有一个有意留给创建窗体的应用程序使用的 32 位的值）。

在设置为透明窗体时，需先使用 GetWindowLong 函数得到扩展的窗体风格（GWL_EXSTYLE）。

3．SetWindowLong函数

与 GetWindowLong 函数对应，SetWindowLong 函数用来修改给定窗体的一个属性。该函数的原型如下：

```
Private Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" _
    (ByVal hWnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As
Long
```

该函数前两个参数与 GetWindowLong 相同，最后的参数 dwNewLong 为修改窗体的属性值。

4．制作透明窗体的过程

了解以上 API 函数后，就可以在 VBE 中开始制作透明窗体了。具体步骤如下：

（1）在 Excel 中按快捷键 Alt+F11 进入 VBE。

（2）单击菜单【插入】|【用户窗体】命令，增加一个用户窗体。向用户窗体中添加 1 个按钮，设置用户窗体及按钮控件的属性，得到如图 29-5 所示窗体。

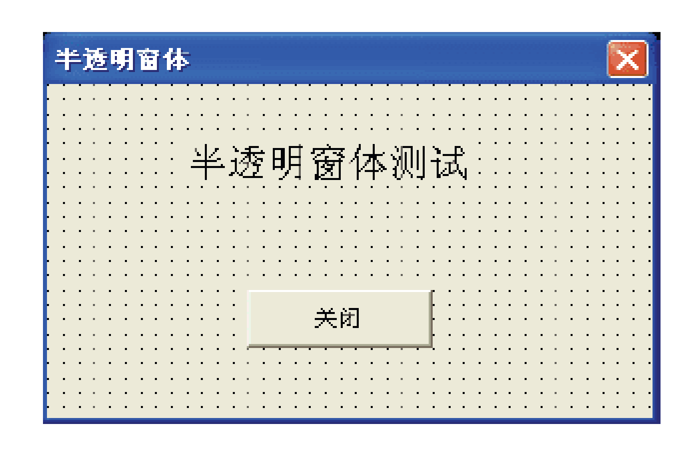


图 29-5　用户窗体

（3）双击窗体空白处打开代码窗体，在声明部分粘贴以下 4 个 API 函数的定义。

```
Private Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" _
    (ByVal hWnd As Long, ByVal nIndex As Long) As Long

Private Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" _
    (ByVal hWnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As
Long

Private Declare Function SetLayeredWindowAttributes Lib "user32" _
    (ByVal hWnd As Long, ByVal crKey As Long, ByVal bAlpha As Byte, _
    ByVal dwFlags As Long) As Long

Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" ( _
    ByVal lpClassName As Long, ByVal lpWindowName As String) As Long
```

（4）在以上 API 函数中，将使用部分常数，也需要在代码的声明部分进行定义，具体如下：

```
Private Const WS_EX_LAYERED = &H80000
Private Const GWL_EXSTYLE = (-20)
```

```
Private Const LWA_ALPHA = &H2
Private Const LWA_COLORKEY = &H1
```

（5）在窗体的初始化事件中编写以下代码，调用 API 函数完成透明窗体的设置：

```
Private Sub UserForm_Initialize()
    Dim rtn As Long, hWnd As Long
    hWnd = FindWindow(0&, Me.Caption)        '得到当前窗体的句柄
    rtn = GetWindowLong(hWnd, GWL_EXSTYLE)   '获取扩展属性
    rtn = rtn Or WS_EX_LAYERED
    SetWindowLong hWnd, GWL_EXSTYLE, rtn     '设置扩展属性
    SetLayeredWindowAttributes hWnd, 0, 100, LWA_ALPHA '设置为透明窗体
End Sub
```

在 SetLayeredWindowAttributes 函数中，设置参数 bAlpha 为 0，窗体将完全透明，处于看不见的状态。设置参数 bAlpha 为 255，窗体将不透明（与正常窗体一样）。

（6）关闭代码窗体，执行用户窗体，将得到如图 29-6 所示的半透明窗体效果。窗体处于半透明状，并将下方工作表的线条也显示出来了。

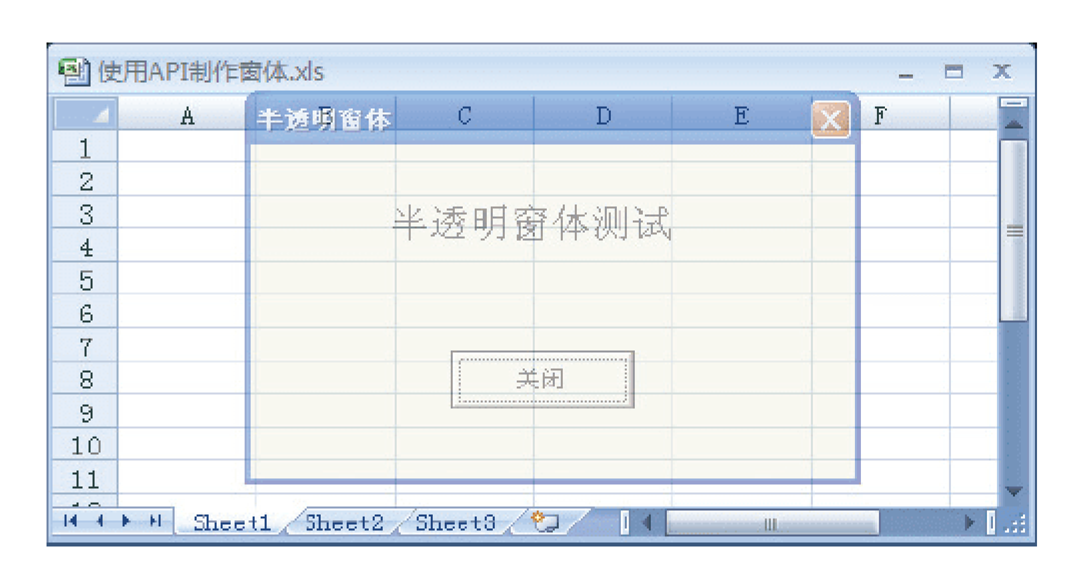


图 29-6　半透明窗体

29.3.2　制作椭圆窗体

正常情况下，用户窗体为矩形。使用 API 函数也可制作如图 29-7 所示的椭圆窗体。本节将演示制作这种椭圆窗体的方法。



图 29-7　椭圆窗体

1. CreateEllipticRgn函数

该函数将创建一个椭圆，函数原型如下：

```
Private Declare Function CreateEllipticRgn Lib "gdi32" ( _
```

```
    ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long)
As Long
```

该函数的 4 个参数用来设置 2 个坐标位置，创建的椭圆与这两个坐标位置构成矩形内切。返回值为创建区域的句柄，供其他函数调用。

提示：该函数创建的区域在不用时一定要用 DeleteObject 函数删除。

2．DeleteObject函数

用该函数删除 GDI 对象（例如画笔、刷子、字体、位图、区域以及调色板等），对象使用的所有系统资源都会被释放。该函数原型如下：

```
Private Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long)
As Long
```

参数 hObject 为一个 GDI 对象的句柄

在程序中用 API 创建一个区域后，该区域用户是看不见的，但它却是一个对象。如果不将其删除，它就会存在于系统中消耗系统资源，所以不用的时候就应该将其删除。

3．SetWindowRgn函数

该函数可将指定窗体设置为指定区域形状，函数原型如下：

```
Private Declare Function SetWindowRgn Lib "user32" ( _
    ByVal hWnd As Long, ByVal hRgn As Long, ByVal bRedraw As Boolean) As Long
```

各参数的含义如下所述。

- hWnd：为将设置区域的窗体；
- hRgn：为将设置区域的句柄，一旦设置了该区域，就不能使用或修改该区域句柄，也不要删除它；
- bRedraw：该函数若为 True，则立即重画窗体。

4．制作椭圆窗体的过程

了解以上 API 函数后，就可以在 VBE 中开始制作椭圆窗体了。具体步骤如下：

（1）在 Excel 中按快捷键 Alt+F11 进入 VBE。

（2）单击菜单【插入】|【用户窗体】命令，增加一个用户窗体。向用户窗体中添加 2 个标签、2 个文字框、2 个按钮，设置用户窗体及控件的属性，得到如图 29-8 所示窗体。

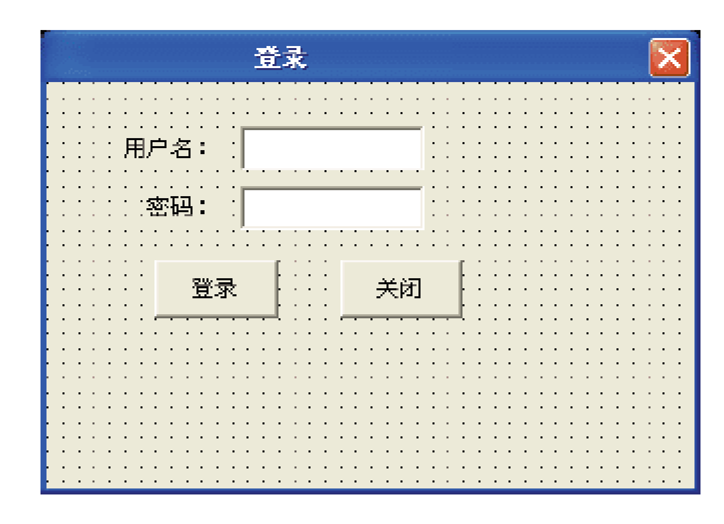


图 29-8 【登录】窗体

提示：控件放置的位置、窗体的大小将影响椭圆窗体显示的效果，可在调试程序时逐步调整控件的位置和窗体的大小，得到满意的效果。

（3）双击窗体空白处打开代码窗体，在声明部分粘贴以下 4 个 API 函数的定义：

```
Private Declare Function CreateEllipticRgn Lib "gdi32" ( _
    ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long)
    As Long

Private Declare Function SetWindowRgn Lib "user32" ( _
    ByVal hWnd As Long, ByVal hRgn As Long, ByVal bRedraw As Boolean) As Long

Private Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long)
As Long

Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" ( _
    ByVal lpClassName As Long, ByVal lpWindowName As String) As Long
```

（4）在窗体的初始化事件中编写以下代码，调用 API 函数完成椭圆窗体的设置：

```
Private Sub UserForm_Initialize()
    Dim rgn As Long, hWnd As Long
    hWnd = FindWindow(0&, Me.Caption)                    '获得窗体的句柄
    rgn = CreateEllipticRgn(0, 0, Me.Width, Me.Height)   '创建椭圆
    SetWindowRgn hWnd, rgn, True                         '改变窗体的区域
    Call DeleteObject(Rng)                               '删除 GDI 对象
End Sub
```

以上代码首先调用 CreateEllipticRgn 函数，以当前窗体矩形大小为参数创建一个椭圆，再使用 SetWindowRgn 函数将当前窗体设置为椭圆，最后删除 CreateEllipticRgn 函数创建的区域。

（5）关闭代码窗体，执行用户窗体，即可得到如图 29-7 所示的椭圆窗体。

29.3.3　制作不规则窗体

制作椭圆窗体时，首先绘制一个椭圆区域，再使用 SetWindowRgn 函数将窗体剪切为该区域样式。同理，如果要制作不规则窗体，则需要首先绘制一个不规则的区域，再使用 SetWindowRgn 函数进行设置即可。

要绘制不规则区域，可使用 API 函数 CreatePolygonRgn，该函数原型如下：

```
Private Declare Function CreatePolygonRgn Lib "gdi32" ( _
    lpPoint As POINTAPI, ByVal nCount As Long, ByVal nPolyFillMode As Long)
As Long
```

该函数创建一个由一系列点围成的区域。Windows 在需要时自动将最后点与第一点相连以封闭多边形。

各参数的含义如下所述。

- lpPoint：为 POINTAPI 类型的变量。POINTAPI 是一个用于描述点坐标的结构，有

两个成员 x 与 y。可以在【API 浏览器】窗口中找到它。在 VBA 中需要定义一个数组，并将这个数组的第一个元素作为这里的参数。

- nCount：多边形的点数。
- nPolyFillMode：描述多边形填充模式。可为 ALTERNATE 或 WINDING 常数。

其他 API 函数在前面的例子中都已介绍过，这里不再重复介绍。制作不规则窗体的具体步骤如下：

（1）在 Excel 中按快捷键 Alt+F11 进入 VBE。

（2）单击菜单【插入】|【用户窗体】命令，增加一个用户窗体。向用户窗体中添加 1 个按钮，设置用户窗体及控件的属性，得到如图 29-9 所示窗体。



图 29-9　原始的【不规则窗体】对话框

（3）双击窗体空白处打开代码窗体，在声明部分粘贴以下 4 个 API 函数的定义：

```
Private Declare Function SetWindowRgn Lib "user32" ( _
    ByVal hWnd As Long, ByVal hRgn As Long, ByVal bRedraw As Boolean) As Long

Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" ( _
    ByVal lpClassName As Long, ByVal lpWindowName As String) As Long

Private Declare Function CreatePolygonRgn Lib "gdi32" ( _
    lpPoint As POINTAPI, ByVal nCount As Long, ByVal nPolyFillMode As Long)
    As Long

Private Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long)
As Long
```

（4）在 CreatePolygonRgn 函数中要使用到 POINTAPI 结构，在 VBA 中可通过自定义类型来表示。在声明部分编写以下代码创建该自定义类型：

```
Private Type POINTAPI
    x As Long
    y As Long
End Type
```

（5）在窗体的初始化（Initialize）事件中编写代码，定义不规则形状各点的坐标，再使用这些坐标数组绘制一个多边形，SetWindowRgn 函数使用该多边形设置窗体的外形，最后删除多边形区域。具体代码如下：

```
Private Sub UserForm_Initialize()
```

```
    Dim hWnd As Long, Result As Long
    Dim P(5) As POINTAPI                              '定义自定义类型数组
    P(0).x = 0                                        '设置区域坐标点
    P(0).y = 0
    P(1).x = Me.Width
    P(1).y = 0
    P(2).x = Me.Width - 50
    P(2).y = Me.Height - 50
    P(3).x = Me.Width - 100
    P(3).y = Me.Height
    P(4).x = 0
    P(4).y = Me.Height / 2
    Result = CreatePolygonRgn(P(0), 5, 1)             '创建区域
    hWnd = FindWindow(0&, Me.Caption)                 '查找窗体句柄
    SetWindowRgn hWnd, Result, True                   '改变窗体区域
    Call DeleteObject(Result)                         '删除区域
End Sub
```

（6）关闭代码窗体，执行用户窗体，即可得到如图 29-10 所示的不规则窗体。



图 29-10　创建好的【不规则窗体】对话框

提示：在 VBE 中调整窗体的原始大小，可得到形状不同的不规则窗体。

29.4　获取系统信息

在 VBA 中没有操作系统信息的相关命令，这时可使用 Windows API 函数来获取系统的相关信息。

29.4.1　获取内存状态

使用 GlobalMemoryStatus 函数可获得当前可用的物理和虚拟内存信息，该函数原型如下：

```
Public Declare Sub GlobalMemoryStatus Lib "kernel32" (lpBuffer As
MEMORYSTATUS)
```

参数 lpBuffer 类型为 MEMORYSTATUS 结构，在 VBA 中可通过自定义类型设置该结

构。函数的返回信息会被存储在 MEMORYSTATUS 结构中。

类型 MEMORYSTATUS 的定义如下：

```
Public Type MEMORYSTATUS
       dwLength As Long
       dwMemoryLoad As Long
       dwTotalPhys As Long
       dwAvailPhys As Long
       dwTotalPageFile As Long
       dwAvailPageFile As Long
       dwTotalVirtual As Long
       dwAvailVirtual As Long
End Type
```

类型中各成员的含义如下所述。

- dwLength：MEMORYSTATUS 结构的大小，在调用 GlobalMemoryStatus 函数前用 Len 函数求得，用来供函数检测结构的版本。
- dwMemoryLoad：返回一个介于 0～100 之间的值，用来指示当前系统内存的使用率。
- dwTotalPhys：返回总的物理内存大小，以字节（B）为单位。
- dwAvailPhys：返回可用的物理内存大小，以字节（B）为单位。
- dwTotalPageFile：显示可以存在页面文件中的字节数。注意这个数值并不表示此页面文件在磁盘上的真实物理大小。
- dwAvailPageFile：返回可用的页面文件大小，以字节（B）为单位。
- dwTotalVirtual：返回调用进程的用户模式部分的全部可用虚拟地址空间，以字节（B）为单位。
- dwAvailVirtual：返回调用进程的用户模式部分的实际自由可用的虚拟地址空间，以字节（b）为单位。

了解 GlobalMemoryStatus 函数以后，就可以在 Excel 中编写函数来获取内存的状态了。具体步骤如下：

（1）在 Excel 中按快捷键 Alt+F11 进入 VBE。

（2）单击菜单【插入】|【模块】命令，向工程中插入一个模块体。

（3）在模块的声明部分粘贴 API 函数的定义如下：

```
Public Declare Sub GlobalMemoryStatus Lib "kernel32" (lpBuffer As
MEMORYSTATUS)
```

（4）在模块的声明部分创建自定义类型，并定义一个模块变量为该自定义类型，用来保存返回的系统信息。具体代码如下：

```
Public Type MEMORYSTATUS
    dwLength As Long
    dwMemoryLoad As Long
    dwTotalPhys As Long
    dwAvailPhys As Long
    dwTotalPageFile As Long
```

```
    dwAvailPageFile As Long
    dwTotalVirtual As Long
    dwAvailVirtual As Long
End Type
Dim MemStat As MEMORYSTATUS
```

（5）接着编写以下过程，通过 GlobalMemoryStatus 函数获取系统信息，再从自定义类型中获取与内存相关的数据。具体代码如下：

```
Private Sub 内存信息()
    Dim str1 As String, temp As Single
    MemStat.dwLength = Len(MemStat)
    GlobalMemoryStatus MemStat

    temp = Round(MemStat.dwTotalPageFile / 1024 / 1024, 2)
    str1 = "物理内存: " & temp & "MB" & vbNewLine
    temp = Round(MemStat.dwAvailPhys / 1024 / 1024, 2)
    str1 = str1 & "可用内存: " & temp & "MB" & vbNewLine
    temp = Round(MemStat.dwTotalPageFile / 1024 / 1024, 2)
    str1 = str1 & "虚拟内存: " & temp & "MB" & vbNewLine
    temp = Round(MemStat.dwAvailPageFile / 1024 / 1024, 2)
    str1 = str1 & "可用虚拟内存: " & temp & "MB"
    MsgBox str1, vbOKOnly, "内存信息"
End Sub
```

提示： GlobalMemoryStatus 函数返回的内存大小是以字节计算的，将其除以两次 1024，转换为以兆字节（MB）为单位。

执行以上过程，将弹出如图 29-11 所示的对话框，显示了当前计算机系统的内存信息。



图 29-11 【内存信息】对话框

29.4.2　获取键盘信息

在 VBA 中，没有提供获取键盘状态的函数。在程序中若需要查询键盘状态，可编写代码调用 API 函数节器完成。

1．GetKeyState函数

使用该函数可获取键盘锁定键的状态，函数原型如下：

```
Public Declare Function GetKeyState Lib "user32" (ByVal nVirtKey As Long)
```

```
As Integer
```

参数 nVirtKey 为需要获取状态的键常数，在 VBA 中，每个键都有一个对应的常数，例如，CapsLock 键的常数为 vbKeyCapital。

该函数返回值为一个整型值，如果最低位为 1，则表示锁定键是打开的，因此可以使用以下代码来判断：

```
CBool(GetKeyState(vbKeyCapital) And 1)
```

将 GetKeyState 函数的返回值与整数 1 进行逻辑与运算，来取得返回值的最低位，然后通过 CBool 函数将其转换为逻辑值，即可得到指定锁定键的状态。

2. GetKeyboardState函数和SetKeyboardState函数

设置锁定键状态需使用 GetKeyboardState 和 SetKeyboardState 函数来操作。这两个函数都用一个 256 字节缓冲区取得或设置 256 个键设置，在 VBA 中设置一个有 256 个元素的字节数组来作为缓冲区，这个数组记录着键盘状态，每个键在数组中的位置由 VK_常数决定。

一旦用 GetKeyboardState 函数取得键盘各键状态并存储到缓冲区中后，即可修改特定的键设置，然后用 SetKeyboardState 函数存回设置，达到修改键设置的目的。

GetKeyboardState 函数可取得键盘上每个虚拟键当前的状态，其函数原型如下：

```
Declare Function GetKeyboardState Lib "user32" Alias "GetKeyboardState" _
    (pbKeyState As Byte) As Long
```

参数 pbKeyState 为具有 256 个元素的字节数组的第一个项目。

如果函数返回非 0 表示成功，返回 0 则表示失败。

SetKeyboardState 函数用来设置每个虚拟键当前在键盘上的状态，其声明格式如下：

```
Declare Function SetKeyboardState Lib "user32" Alias "SetKeyboardState" _
    (lppbKeyState As Byte) As Long
```

了解设置键盘状态的函数后，就可以在 VBA 中编写代码，用来设置锁定键的状态。具体步骤如下：

（1）在 Excel 中按快捷键 Alt+F11 进入 VBE。

（2）单击菜单【插入】|【模块】命令，向工程中插入一个模块体。

（3）在模块的声明部分粘贴 API 函数的定义如下：

```
Public Declare Function GetKeyState Lib "user32" (ByVal nVirtKey As Long)
As Integer
Public Declare Function GetKeyboardState Lib "user32" (pbKeyState As Byte)
As Long
Public Declare Function SetKeyboardState Lib "user32" (lppbKeyState As Byte)
As Long
```

（4）编写修改锁定键状态的通用子过程，具体代码如下：

```
Sub SetKeyState(intVKey As Integer, bState As Boolean) '修改键盘状态
```

```
    Dim aBuffer(0 To 255) As Byte            '定义数组作为缓冲区
    GetKeyboardState aBuffer(0)              '获取键盘状态放入缓冲区
    aBuffer(intVKey) = CByte(Abs(bState))    '在缓冲区修改指定键的状态
    SetKeyboardState aBuffer(0)              '使用缓冲区修改键的状态
End Sub
```

程序中使用数组的第一个元素作为参数传递给两个 API 函数，由于数组在 VBA 中是连续存放的，所以 API 函数通过第一个元素的地址就可访问到后面的所有元素。

（5）有了通用函数 SetKeyState，就可以很方便地编写修改锁定键状态的代码了。例如，以下代码可修改 CapsLock 键的状态（在大小写之间相互切换）：

```
Sub ModiCapsLock()                                   '大写锁定
    If CBool(GetKeyState(vbKeyCapital) And 1) Then   '获取 CapsLock 原来的状态
        SetKeyState vbKeyCapital, False              '关闭
    Else
        SetKeyState vbKeyCapital, True               '打开
    End If
End Sub
```

程序首先使用 GetKeyState 函数查询 CapsLock 原来的状态，再将其状态进行切换（即如果原来是打开的，就将其关闭；如果原来是关闭的，就将其打开）。

第 30 章　制作应用程序的帮助

用 Excel VBA 开发的应用程序大多面向初级用户，这些用户计算机软件操作的经验很少。因此，对于一个复杂的应用程序，应为用户提供友好的帮助系统，使用户快速熟悉应用程序的使用方法。本章将介绍使用微软公司的 HTML Help Workshop 制作 CHM 格式帮助系统的方法。

30.1　CHM 帮助概述

在 Web 浏览器出现之前，Windows 平台中的应用程序大部分使用基于 WinHelp 的帮助系统。随着 Web 浏览器的出现和发展，带有超级链接的帮助页面成为了主流。

30.1.1　认识 CHM 帮助文件

在 Windows 操作系统中，使用扩展名.chm 来保存这种基于 HTML 格式的帮助系统。这是通过 HTML Help Workshop 制作得到的帮助系统。这种文件格式在网上广为流传，被称为一种电子书籍格式。

例如，Windows XP 中媒体播放器的帮助系统如图 30-1 所示。该帮助系统就是以网络超文本（HTML）格式制作的。



图 30-1　CHM 格式的帮助系统

HTML Help Workshop 的特点在于，它的每一个帮助页面都是一个 Web 页，可以像浏览网站一样容易地阅读 HTML 帮助文件。该帮助系统支持 HTML、ActiveX、Java 脚本语

言，还可将网络图像（扩展名为.jpg、.gif 和.png）嵌入到帮助文件中。

制作 HTML 帮助文件需要使用 HTML Help Workshop 软件，可到微软的官方网站 http://msdn.microsoft.com/en-us/library/ms669985.aspx 下载该软件。

30.1.2　CHM 帮助文件的构成

HTML Help Workshop 的作用是将各 HTML 文件汇总在一起，再编译成独立的 CHM 帮助文件。一个最简单的 CHM 文件应该包含以下几部分：

1．帮助项目文件

HTML 帮助项目文件是一个纯文本文件，扩展名为.hhp。在项目文件中，将帮助系统中所需要的所有元素组织在一起。例如，帮助主题文件、图像文件、索引文件等。

将各种元素添加到项目文件后，即可通过 HTML Help Workshop 将这些内容编译成一个单独的、扩展名为.chm 的帮助文件。

2．帮助主题文件

帮助主题文件是使用 HTML 创建的网页文件，其扩展名为.htm 或.html。

在使用 HTML Help Workshop 之前，应使用网页制作软件（如 Dreamweaver、FrontPage）将 HTML 文件准备好。如果不会制作网页，也可以通过 Word 编辑每个帮助主题的内容，再将其保存为 HTML 格式即可。在创建帮助主题文件时，可在每个网页中创建超链接，在各主题文件之间进行跳转。

注意：因为 Word 中插入的图片保存为网页时，使用的 CSS 在 HTML Help Workshop 中不能被识别，所以生成的 CHM 帮助系统将不能显示图像。

3．目录文件

帮助文件的目录类似于 Windows 资源管理器的左半部分。目录文件包含帮助中的所有项目，而每个目录又包含条目名称、跳转到帮助主题的捷径等内容。当用户在帮助文件的目录页中单击一个条目标题时，与该条目的标题相链接的 HTML 文件将被打开。

4．关键字和索引文件

关键字是用户可能用到的，并与一个或多个帮助主题文件相关联的词。索引文件包含若干关键字。当用户在编译过的帮助文件里单击索引页，并选择一个关键字时，帮助文件将显示与这个关键字有关的帮助主题列表。

30.2　准备帮助主题文件

制作帮助系统最主要的工作就是准备帮助主题文件。然后将这些主题文件通过 HTML

Help Workshop 软件组织在一起，经过编译得到 CHM 帮助文件。

下面演示准备帮助主题文件的步骤：

（1）创建一个文件夹 HTML，将所有的帮助主题文件放在该文件夹中，方便统一管理。

提示： 文件夹名称可由用户随意设置。

（2）在 Word 或其他网页制作软件中制作好一个帮助主题文件，如图 30-2 所示为在 Dreamweaver 中创建的一个帮助主题文件，在该文件中包含文字和图片。

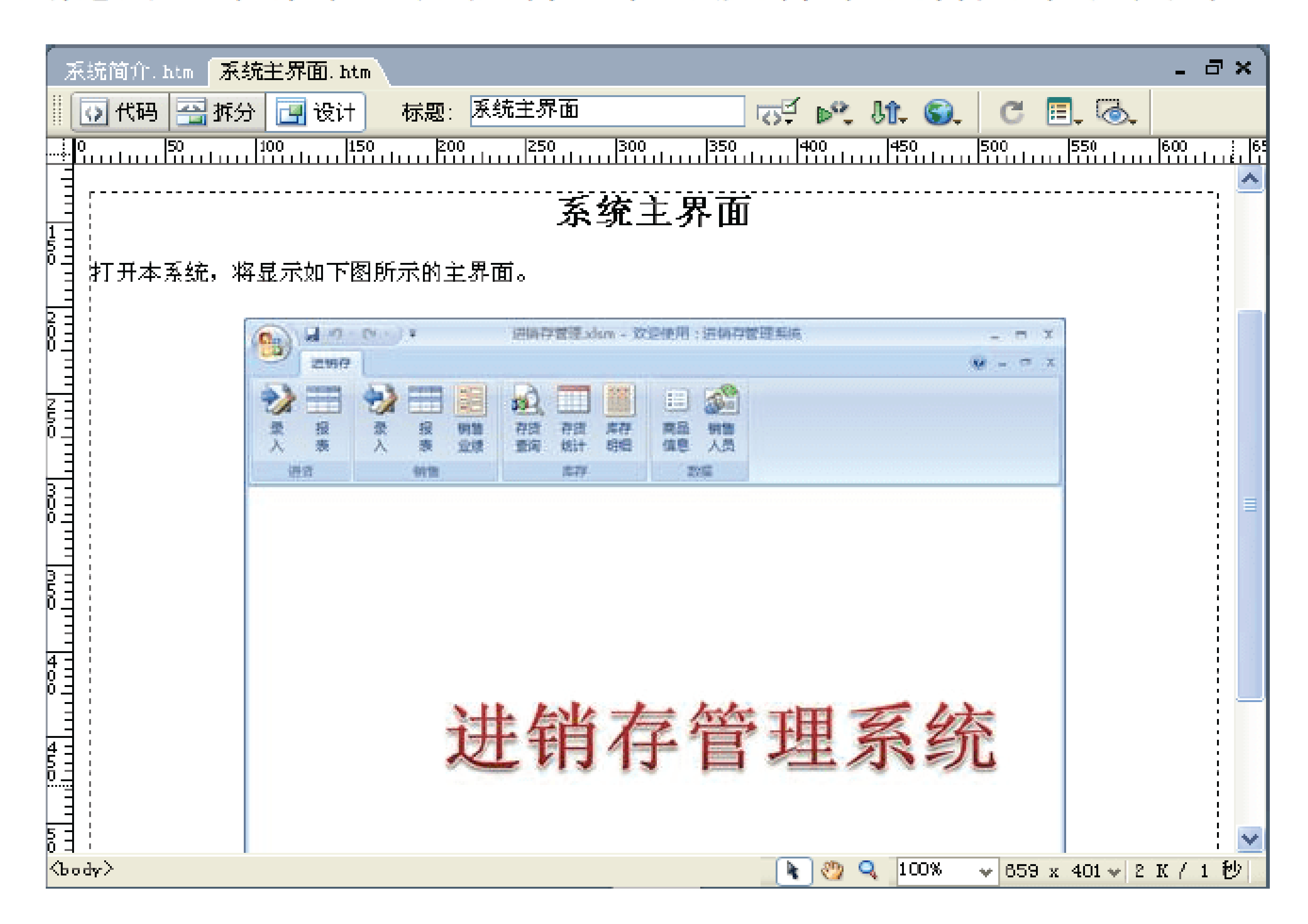


图 30-2　帮助主题文件

（3）在如图 30-3 所示的 HTML 文件中，还可根据需要创建到其他 HTML 页面的超链接。在编译为 CHM 帮助文件中，单击这些超链接也可跳转到指定的页面。

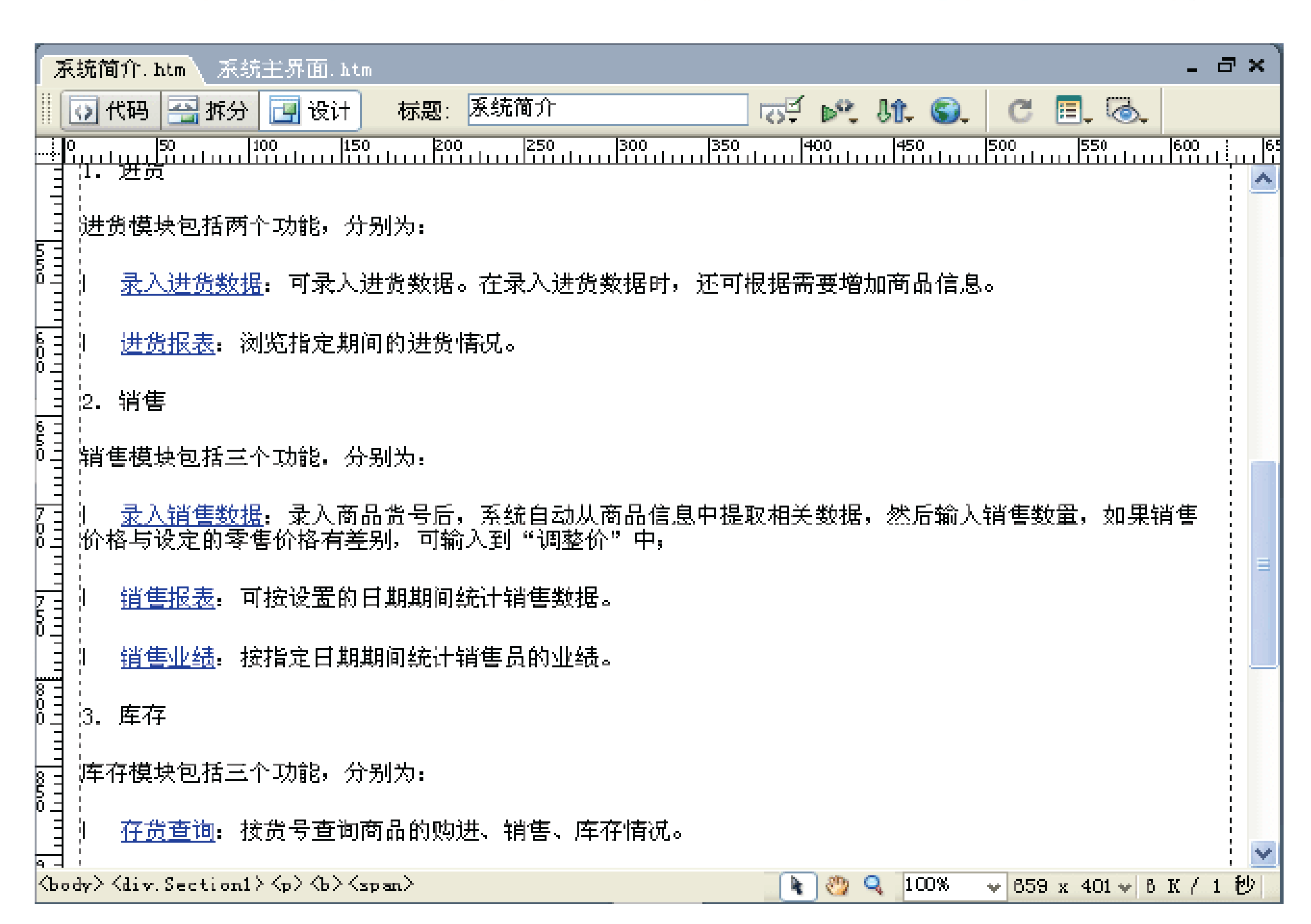


图 30-3　HTML 文件

（4）根据应用程序的功能，分别创建多个不同的 HTML 网页，并将这些网页以方便识别的名称保存到 HTML 文件夹中。

30.3　制作 HTML 帮助系统

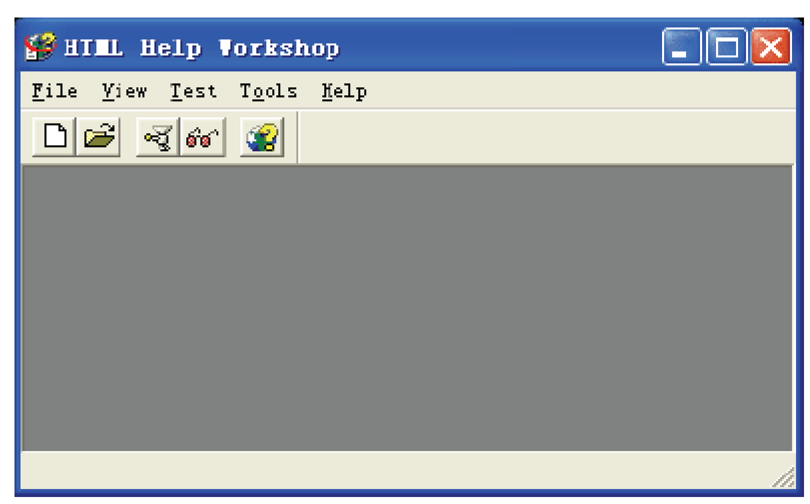
准备好所有的帮助主题文件后，就可以使用 HTML Help Workshop 软件将这些内容组织到一起。本节将介绍 HTML Help Workshop 软件的简单使用方法。

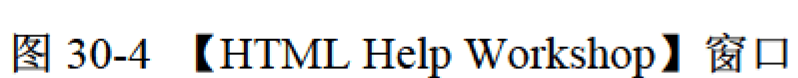
30.3.1　创建项目文件

准备好帮助文件的各主题文件后，就可以通过 HTML Help Workshop 将其编译为一个独立的 CHM 文件了。首先需要创建一个帮助项目文件，具体步骤如下：

（1）启动 HTML Help Workshop，如图 30-4 所示。

（2）单击菜单 File|New 命令，打开 New 对话框，如图 30-5 所示。





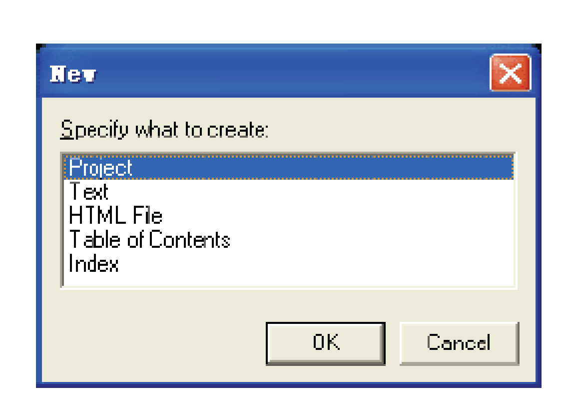


图 30-4 【HTML Help Workshop】窗口　　　图 30-5 【New】对话框

（3）在图 30-5 所示对话框中选择 Project，用来新建一个项目。单击 OK 按钮，进入新建项目向导，如图 30-6 所示。

（4）在向导的第一个对话框中不选中 Convert WinHelp project 复选框，直接单击【下一步】按钮进入如图 30-7 所示对话框，输入放置项目文件的目录及项目文件名称（也可单击 Browse 按钮查找保存项目的文件）。

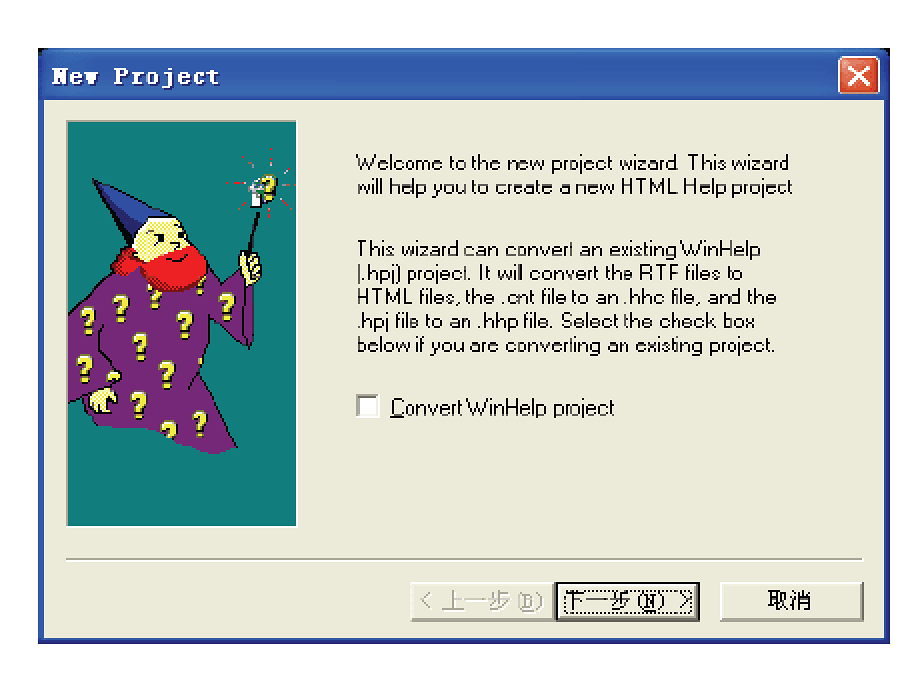




图 30-6　新建项目向导　　　图 30-7　设置项目名称

（5）单击【下一步】按钮打开如图 30-8 所示对话框。

（6）因为已经事先建好了 htm 文件，因此在图 30-8 所示对话框中选中 HTML file 复选框，单击【下一步】按钮将打开如图 30-9 所示的对话框。




图 30-8　设置存在的文件

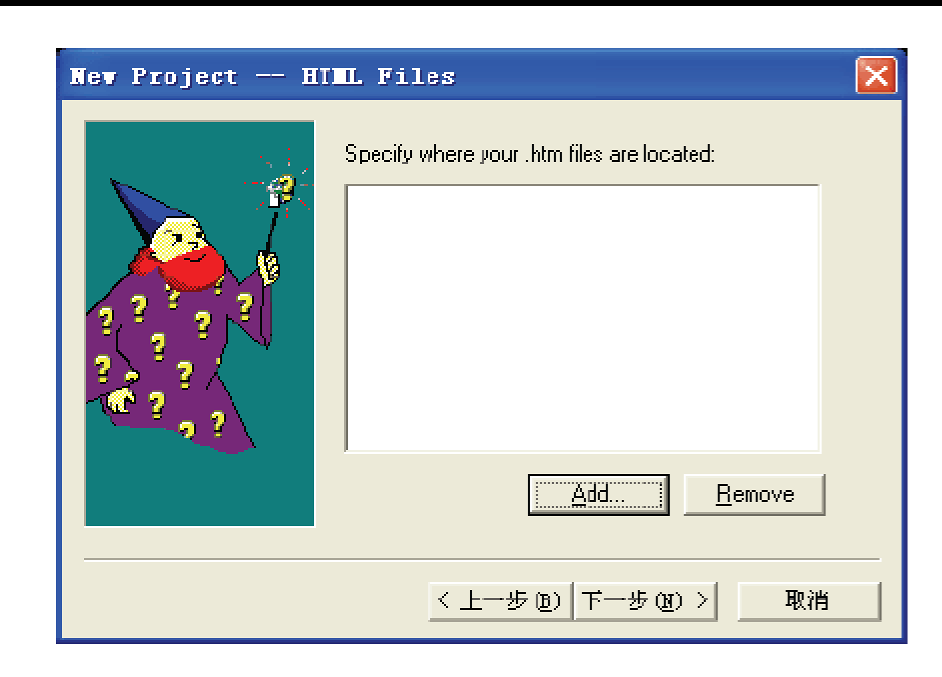


图 30-9　选择 HTML 文件

（7）在图 30-9 所示对话框中单击 Add 按钮，打开如图 30-10 所示【打开】对话框。

（8）选择 30.2 节准备好的帮助主题文件（可一次选择多个文件），单击【打开】按钮，选中的文件将添加至如图 30-11 所示的对话框中。在该对话框中单击 Remove 按钮，可删除列表中选中的文件。

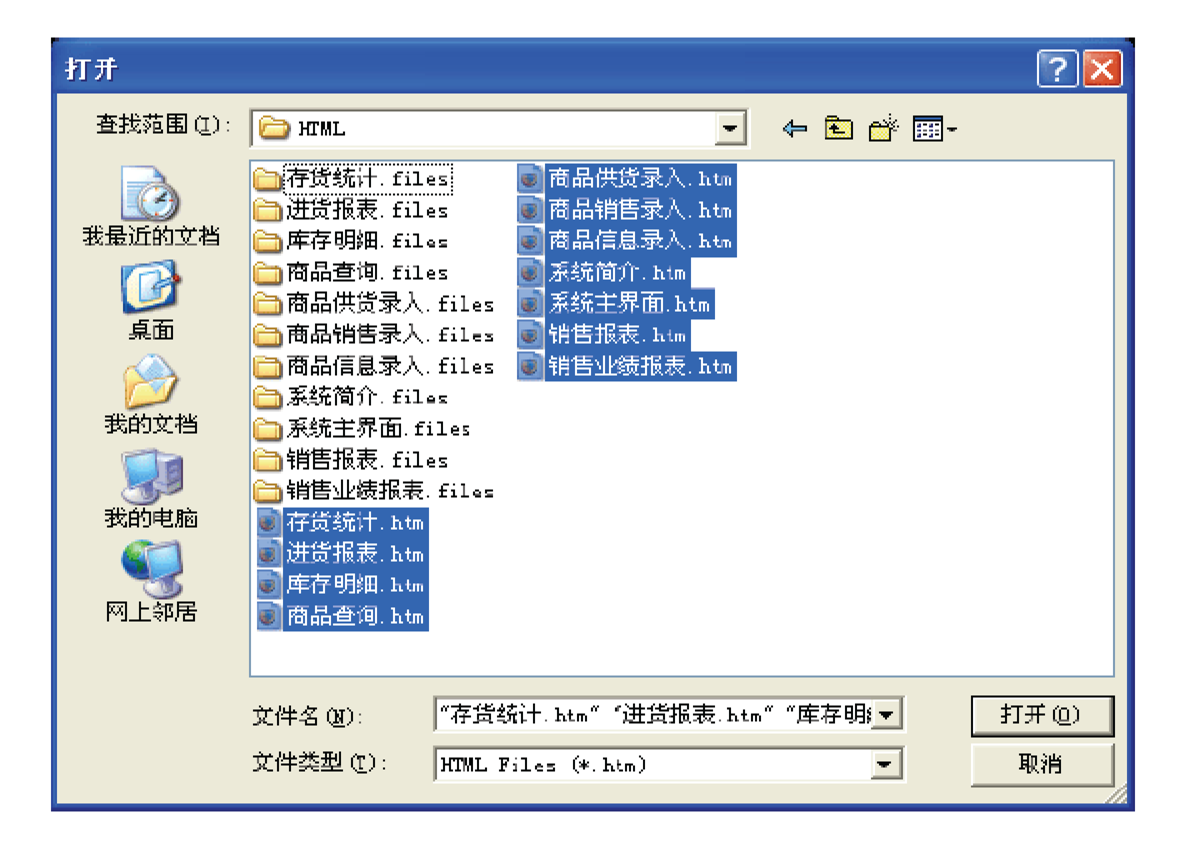


图 30-10　【打开】对话框



图 30-11　添加文件

（9）在图 30-11 所示对话框中单击【下一步】按钮，显示如图 30-12 所示对话框，单击【完成】按钮，新的项目创建完成，然后进入如图 30-13 所示的主界面。



图 30-12　【完成】对话框

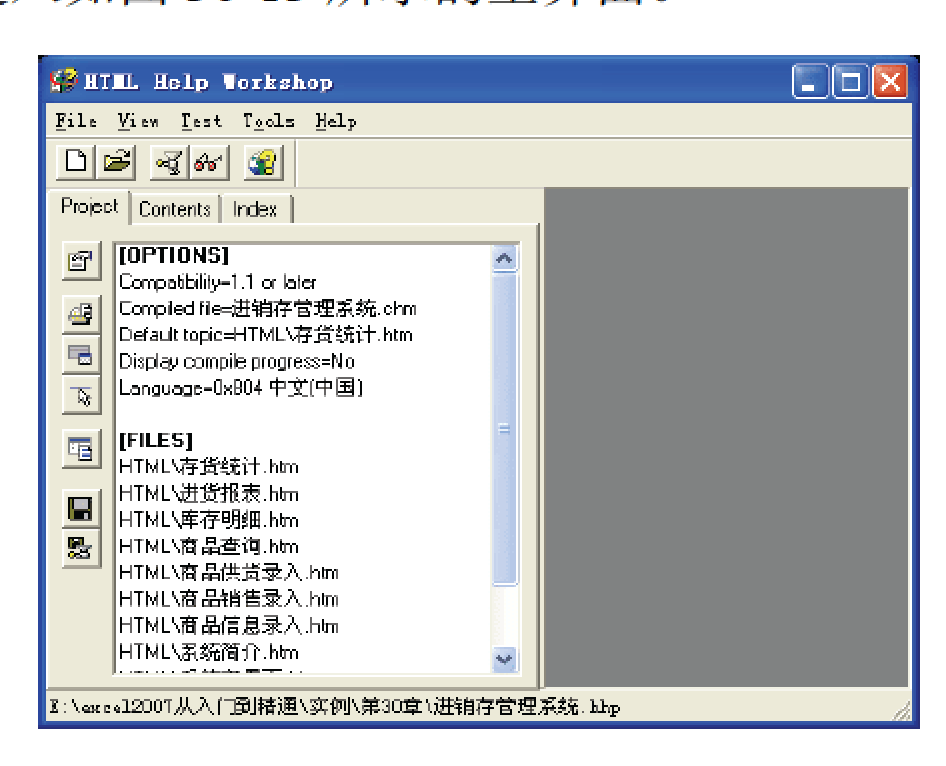


图 30-13　创建的项目

在主界面的左侧有 3 个选项卡，分别是 Project（项目）、Contents（目录）和 Index（索引）。

在 Project 选项卡的左侧有 7 个按钮，可用来进行项目选项设置、添加/删除主题文件、保存项目、编译项目等操作。

30.3.2　创建目录文件

对于帮助主题很多的帮助文件，编辑目录文件是非常关键的工作。目录文件应该包含一个 CHM 文件所有的（目录）主题，而每个目录又包含条目标题（名称）和该条目的主题文件，要避免条目标题与对应的主题不一致的情况出现。创建目标的步骤如下：

（1）在图 30-13 所示窗口中，单击 Contents 选项卡，将打开如图 30-14 所示的对话框，提示项目还没有关联到目录文件（.hhc）。



图 30-14　创建新的目录文件

（2）在对话框中选择 Create a new contents file 单选按钮，并单击 OK 按钮后，打开如图 30-15 所示的【另存为】对话框。

（3）为新目录文件设置名称和存放路径，并单击【保存】按钮将目录文件保存。接着，将进入目录编辑窗口，目录编辑窗口的左侧有 11 个按钮。如图 30-16 所示。

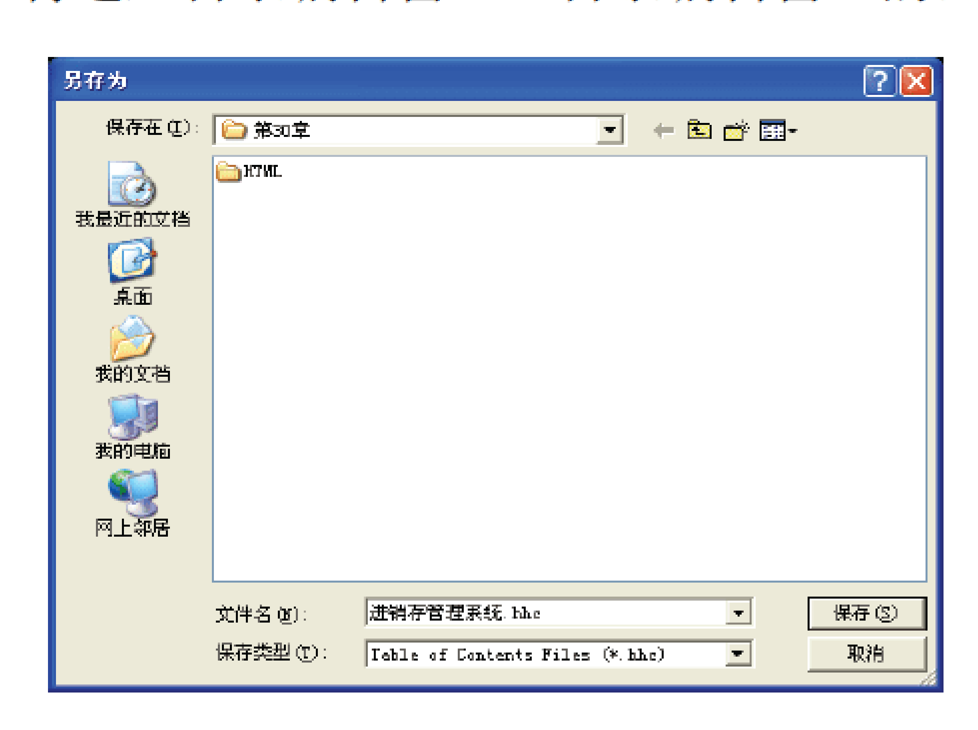



图 30-15　保存目录文件

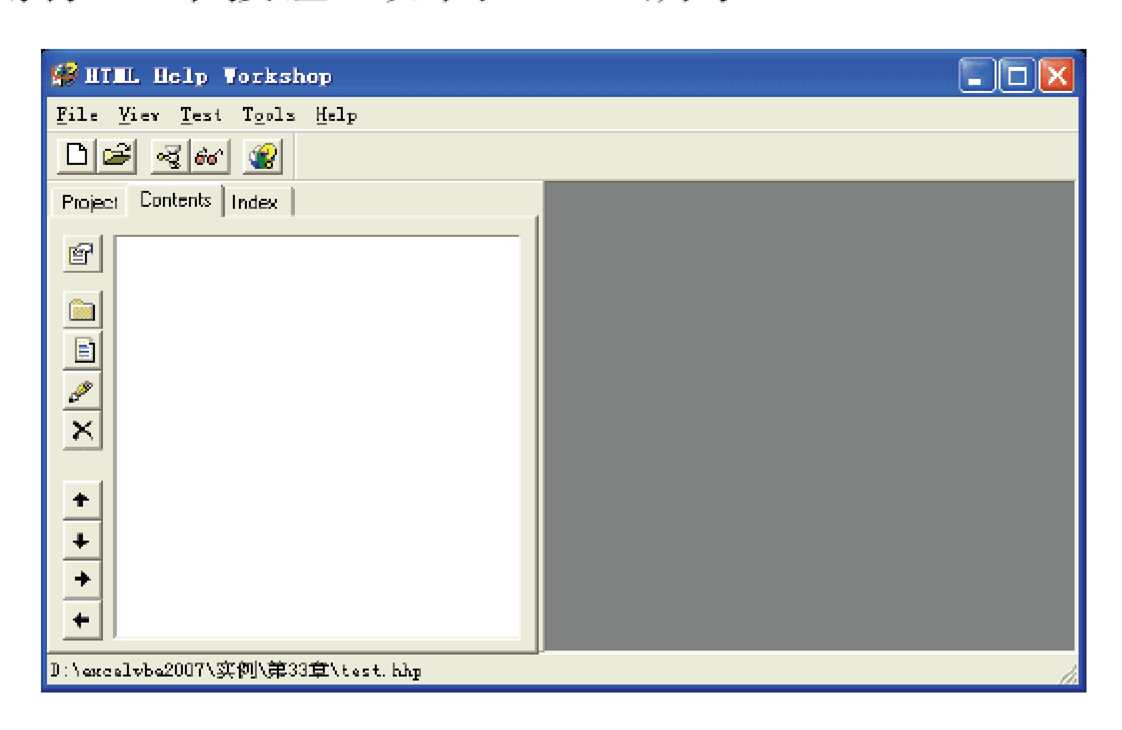


图 30-16　创建目录

（4）单击左侧的 Insert a page（插入页面）按钮（类似资源管理器中的文件），打开如图 30-17 所示的对话框，向项目中增加一个页。

（5）在 Entry title 文本框中输入目录列表中显示的内容（如此处的“系统主界面”），再单击【Add】按钮，打开如图 30-18 所示对话框。

（6）在对话框的 HTML titles 列表框中选择要链接到该目录项的 HTML 文件。也可单击右下方的 Browse 按钮查找未在列表框中出现的 HTML 文件。

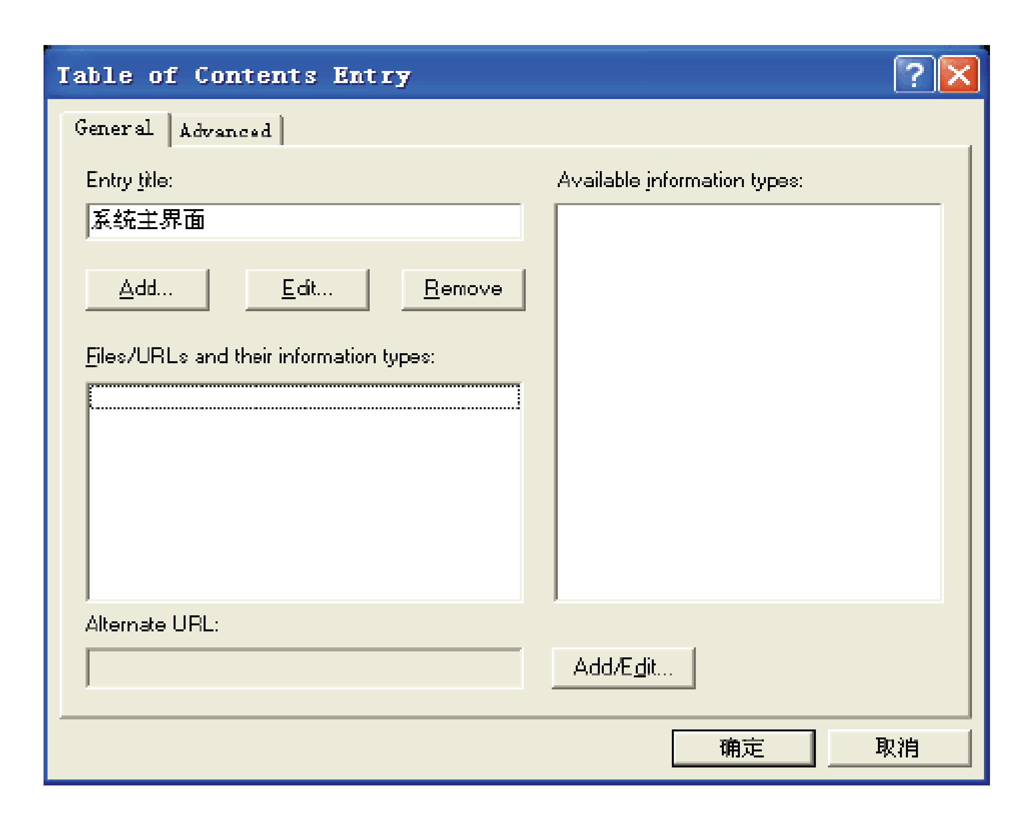


图 30-17　输入目录标题

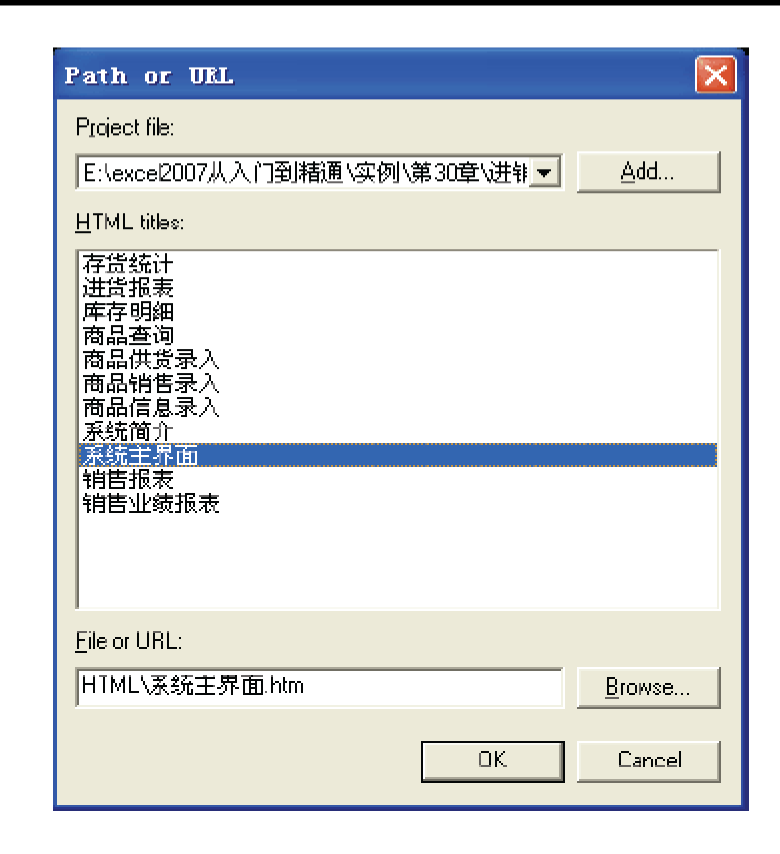


图 30-18　选择主题文件

（7）设置好以后单击 OK 按钮返回上一个对话框，可看到目录项与主题文件已经创建了关联，如图 30-19 所示。

（8）单击【确定】按钮，完成一个目录项的制作。

（9）单击左侧的 Insert a heading（插入标题）按钮（类似资源管理器中的文件夹），可采用相似的方法创建一个标题（标题可不设置对应的 HTML 网页）。使用标题可对不同帮助主题文件进行分组，在标题下面包括目录项。

最后生成的目录结构如图 30-20 所示。

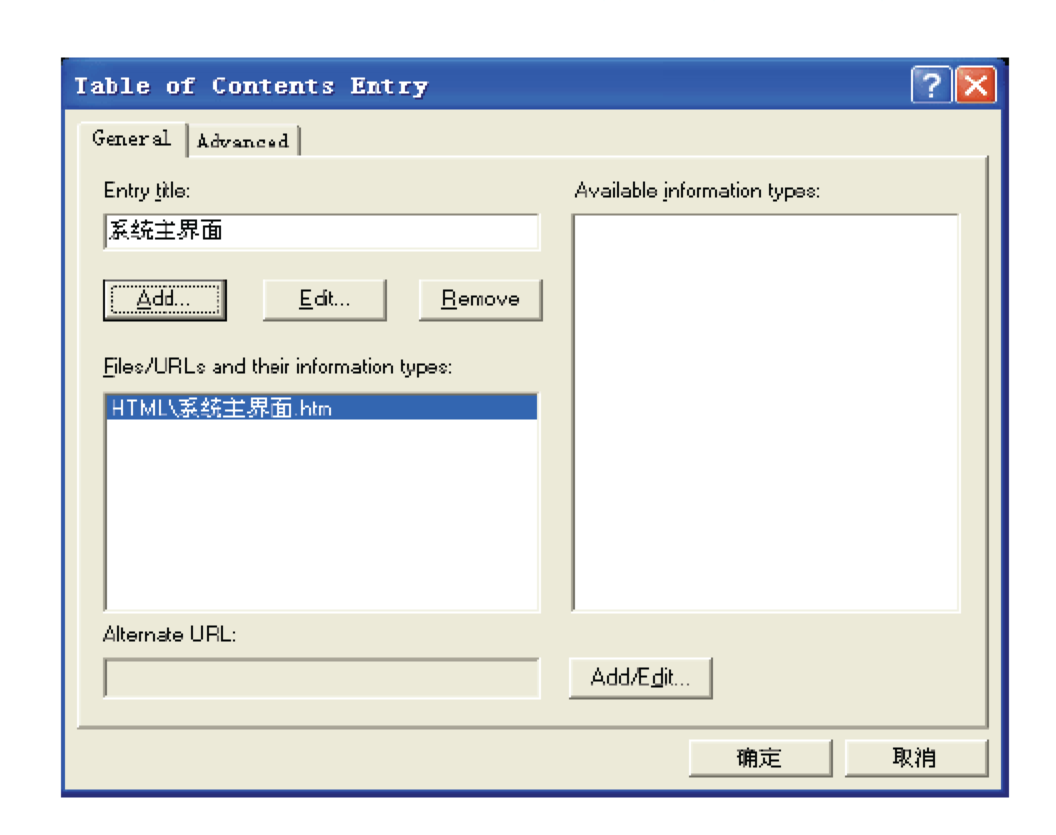


图 30-19　目录与帮助主题的链接

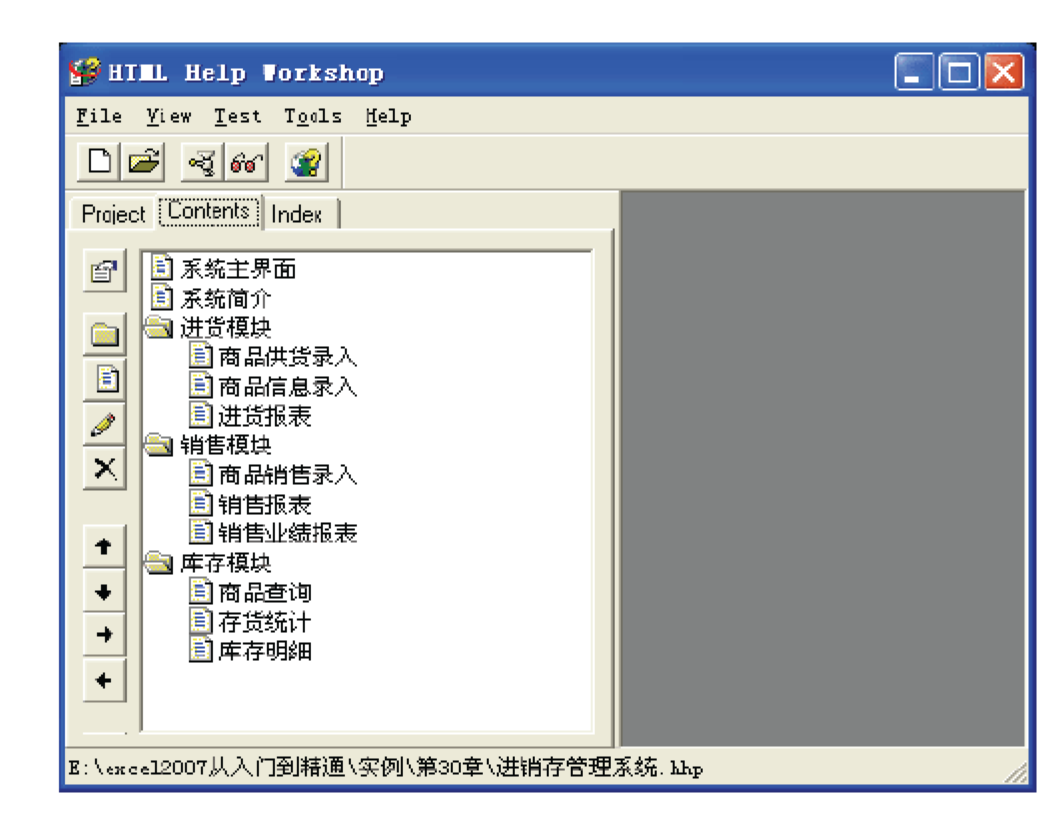


图 30-20　帮助的目录

技巧： 标题可以分为多级，要按照制作的内容统一考虑。如果觉得不满意，可以用左侧的箭头进行调整，也可以选定该条目，单击鼠标右键，不但可以调整，还可以插入标题、主题或目录文件。

30.3.3　创建索引文件

索引文件（hhk）也是一个 HTML 文件，它包含若干个关键词，当用户打开 CHM 文

件后，单击索引标签并输入一个关键词时，CHM 文件将显示与这个关键词有关的主题列表，让用户可以非常方便地找到相关主题。创建索引文件的步骤如下：

（1）在主界面中单击 Index 选项卡，系统将弹出如图 30-21 所示的对话框，提示还没有关联索引文件（.hhk）。

（2）选择 Create a new contents file 单选按钮，并单击 OK 按钮打开如图 30-22 所示的对话框，输入索引文件名称，单击【保存】按钮创建一个空白索引文件。

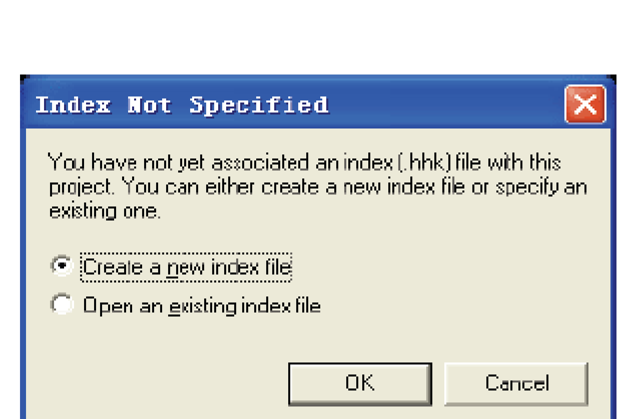


图 30-21　新建索引

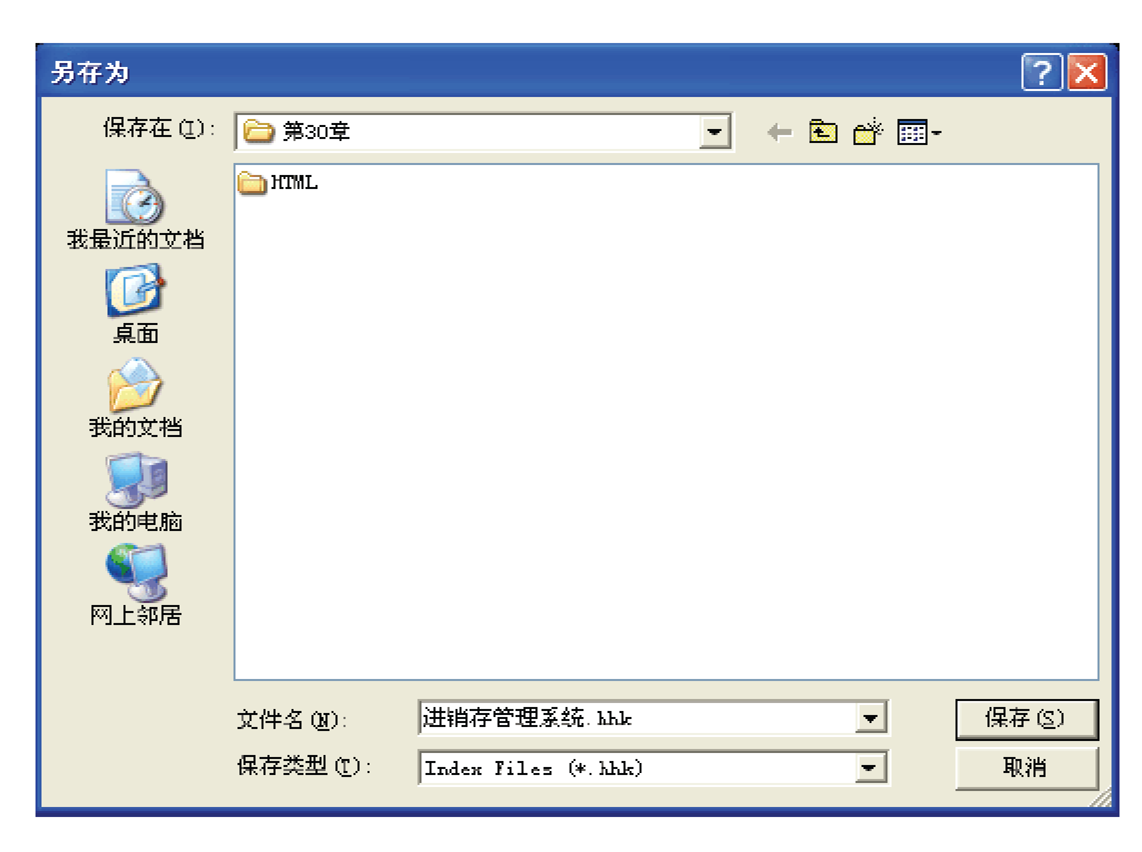


图 30-22　保存索引文件

（3）接着出现索引编辑窗口，索引编辑窗口的左侧有 11 个按钮，如图 30-23 所示。

（4）单击左侧的 Insert a keyWord 按钮，打开 Index Entry 对话框，如图 30-24 所示。在 General 选项卡的 KeyWord 输入框中输入关键词“主界面”。

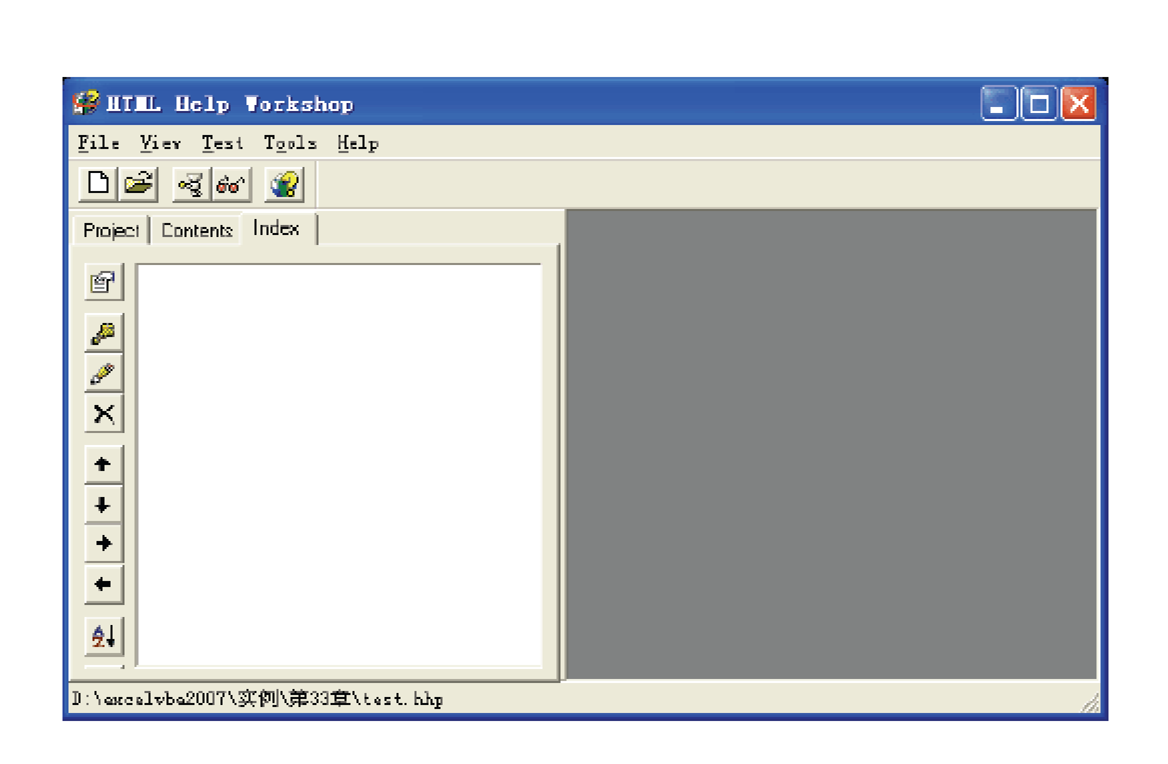


图 30-23　创建索引

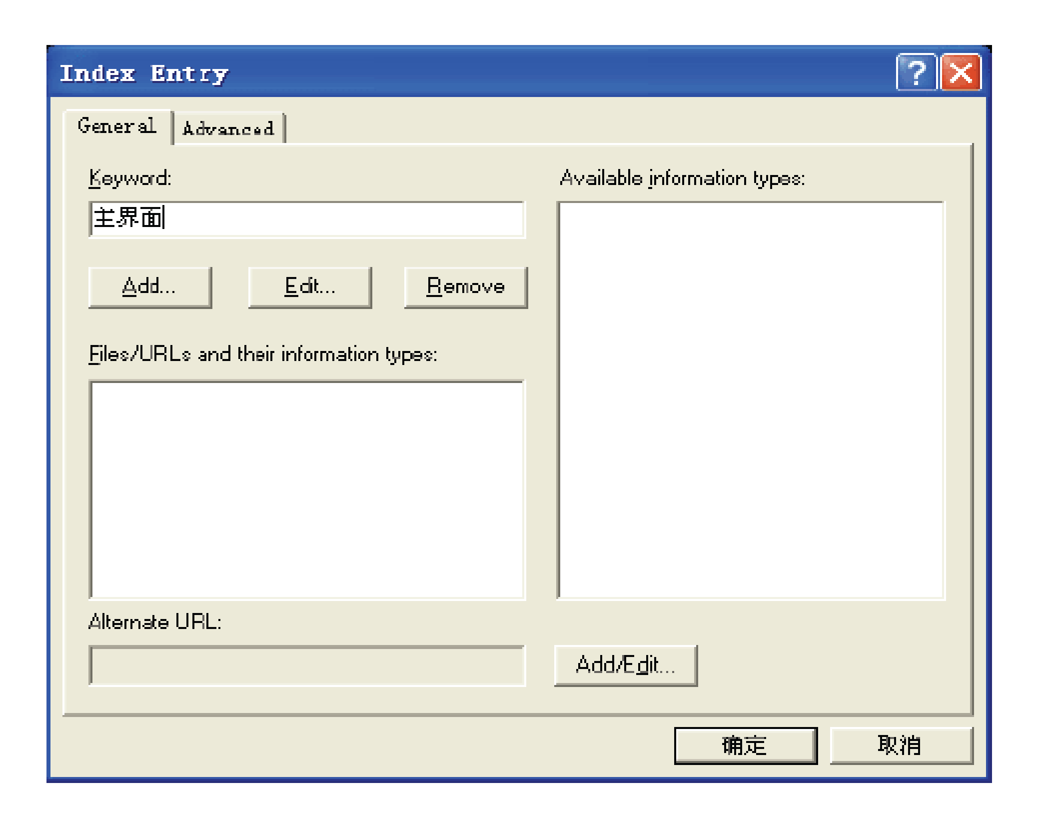
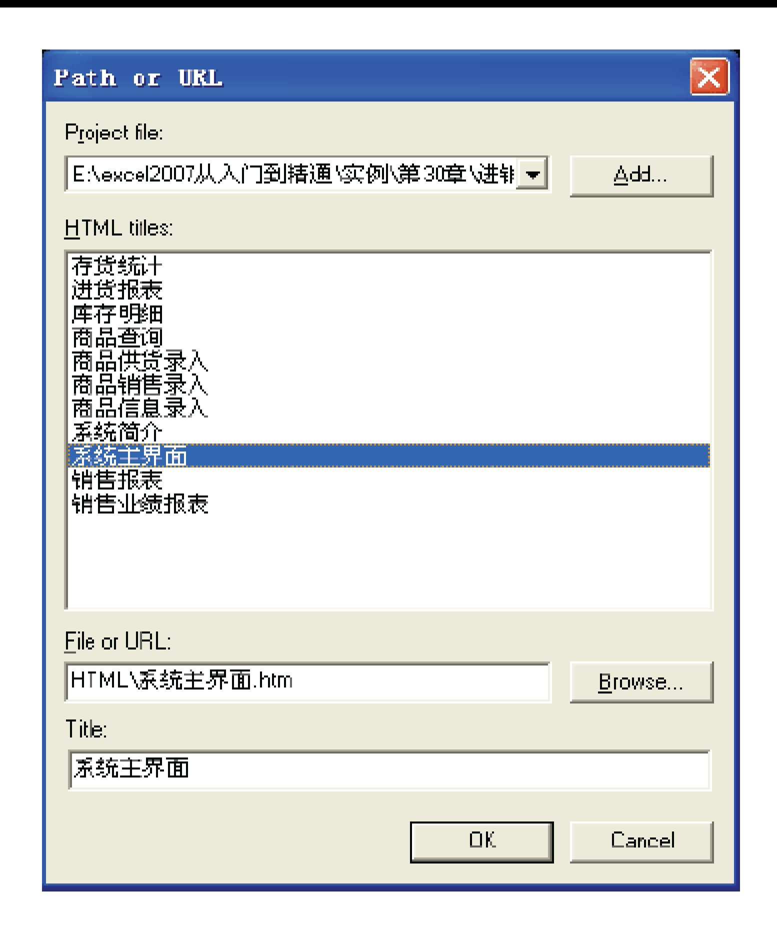


图 30-24　输入关键词

（5）单击下方的 Add 按钮，打开如图 30-25 所示对话框，设置与该关键词相关联的主题文件。然后单击 OK 按钮返回上一个对话框，再单击【确定】按钮完成一个关键字的设置。

（6）用同样的方法可设置其他相关的关键字和主题文件相链接，得到如图 30-26 所示的索引列表。



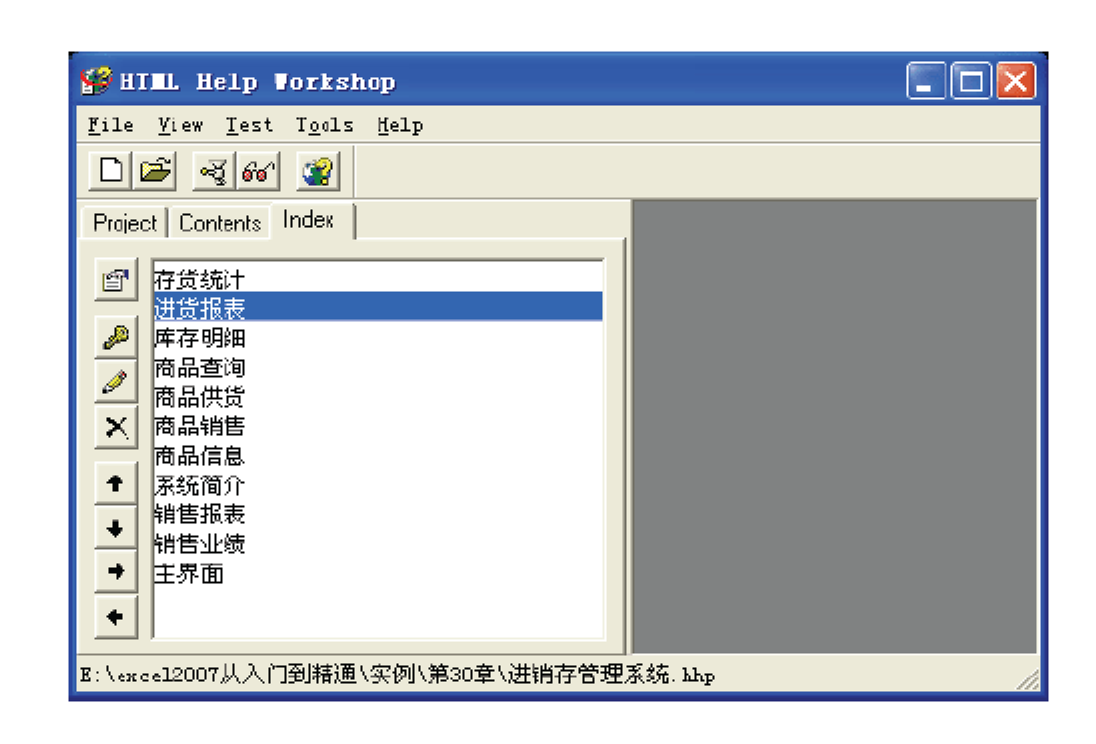



图 30-25　设置索引的链接文件

图 30-26　索引列表

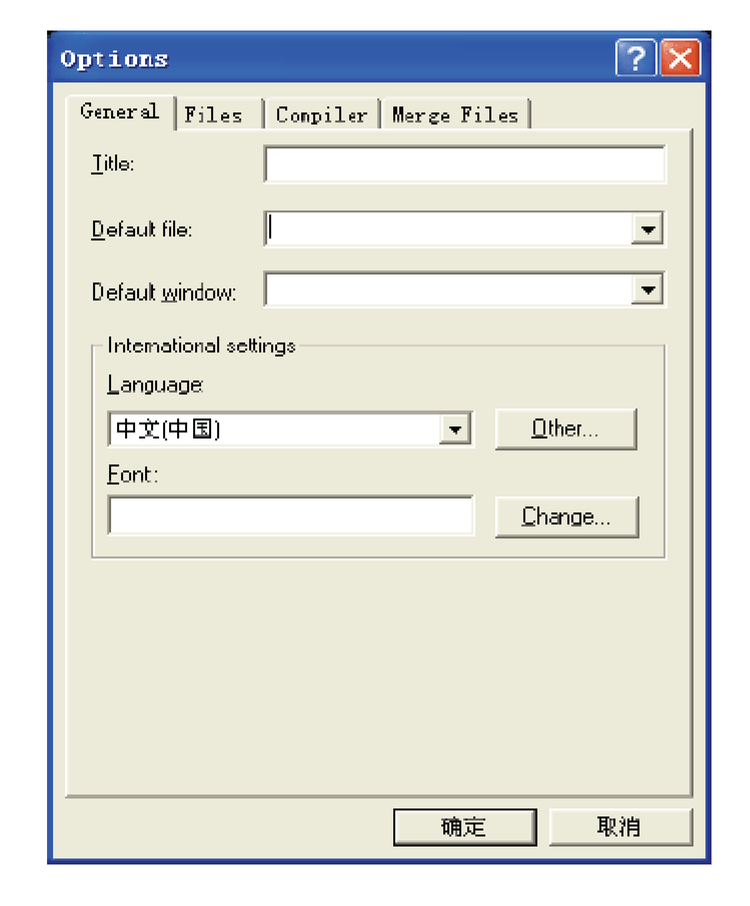
30.3.4　设置帮助文件的选项

帮助文件的选项包括标题、起始页面等多项内容。标题是在帮助窗口标题栏中显示的内容，起始页面是打开帮助文件时首先显示的页面。设置这些选项的步骤如下：

（1）单击 Project 选项卡。

（2）单击左侧的 Change project options 按钮，打开如图 30-27 所示的对话框。

（3）在 Title 文本框中输入标题，在 Default file 列表框中选择起始页（也可手工输入，使用相对路径），如图 30-28 所示。



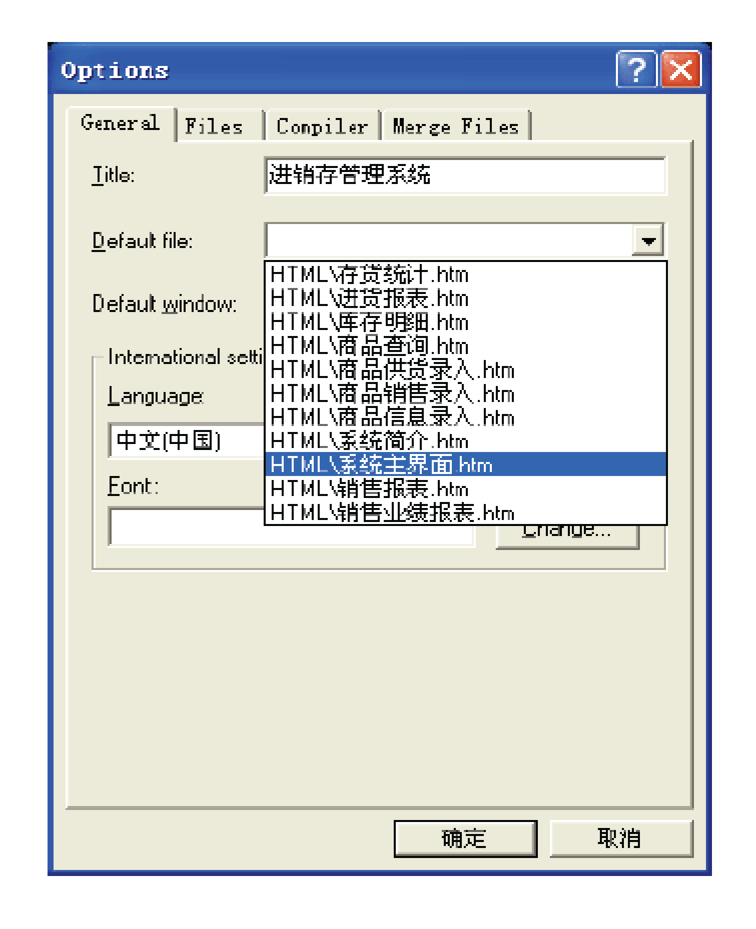


图 30-27　选项

图 30-28　选择起始页

（4）单击 Compiler 选项卡，在下方选中 Compile full-text search information 复选框，使编译生成的帮助文件支持全文搜索（即可搜索帮助主题文件中包含的任意字符），如

图 30-29 所示。

（5）单击【确定】按钮，完成选项参数的设置，回到主窗口中，即可看到设置的参数显示在 OPTIONS 下方，如图 30-30 所示。

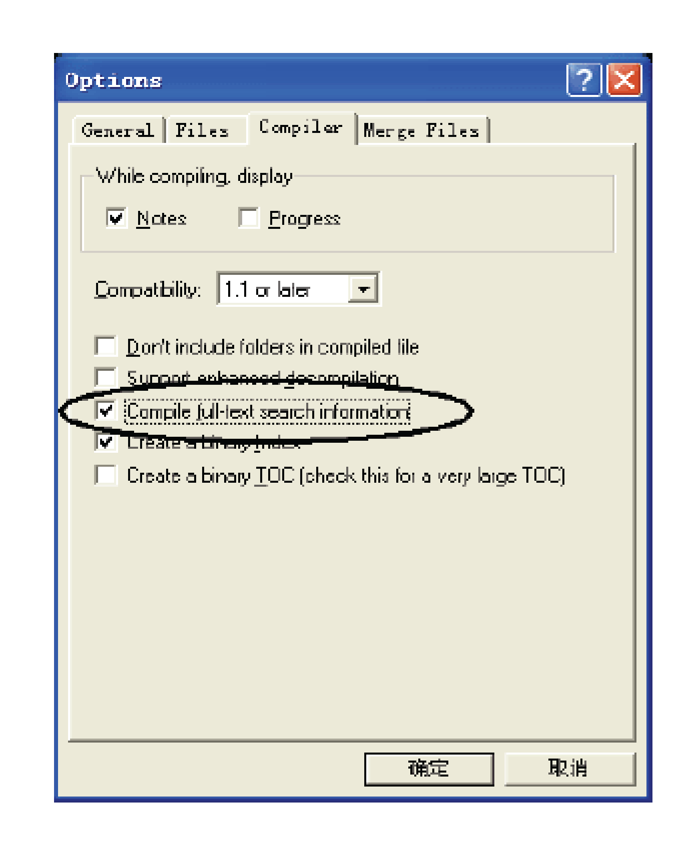


图 30-29　选中全文搜索

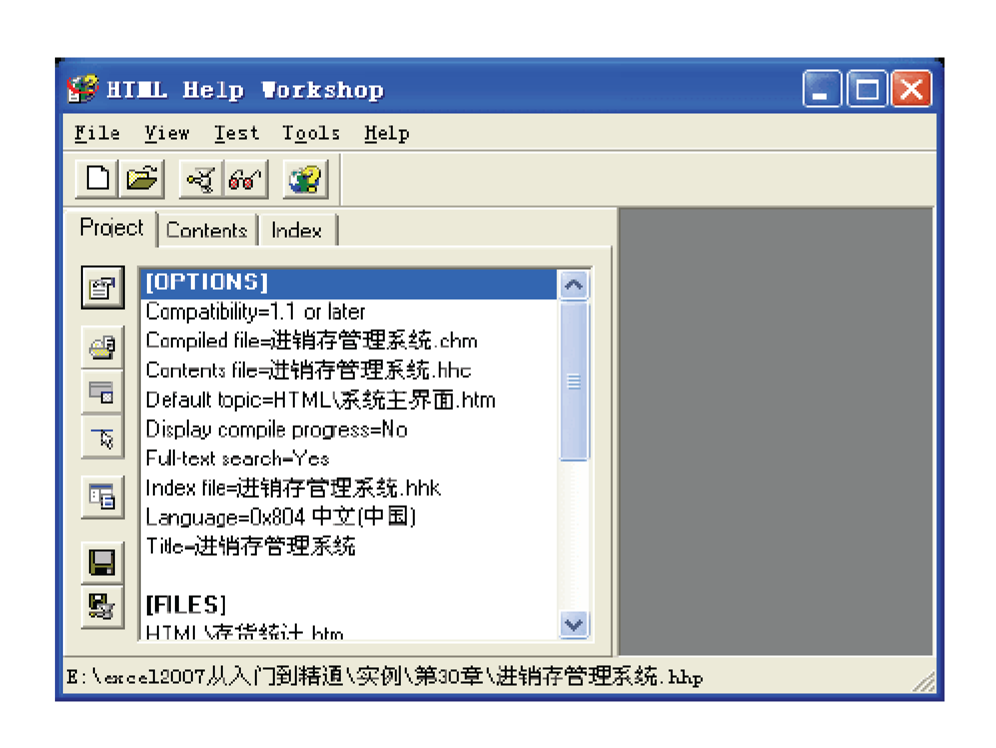


图 30-30　设置的参数

30.3.5　编译生成帮助文件

将目录和索引都设置完成后，帮助文件的大部分工作就都做好了，接着对其进行编译就可生成帮助文件了。编译操作及相关设置步骤如下：

（1）在主界面中选择 Project 选项卡

（2）单击左侧的 Save project file and compile 按钮，将项目文件全部保存，并编译为 CHM 文件。编译过程中，将在主界面右侧的窗口中显示编译的进度及各项提示，如图 30-31 所示。

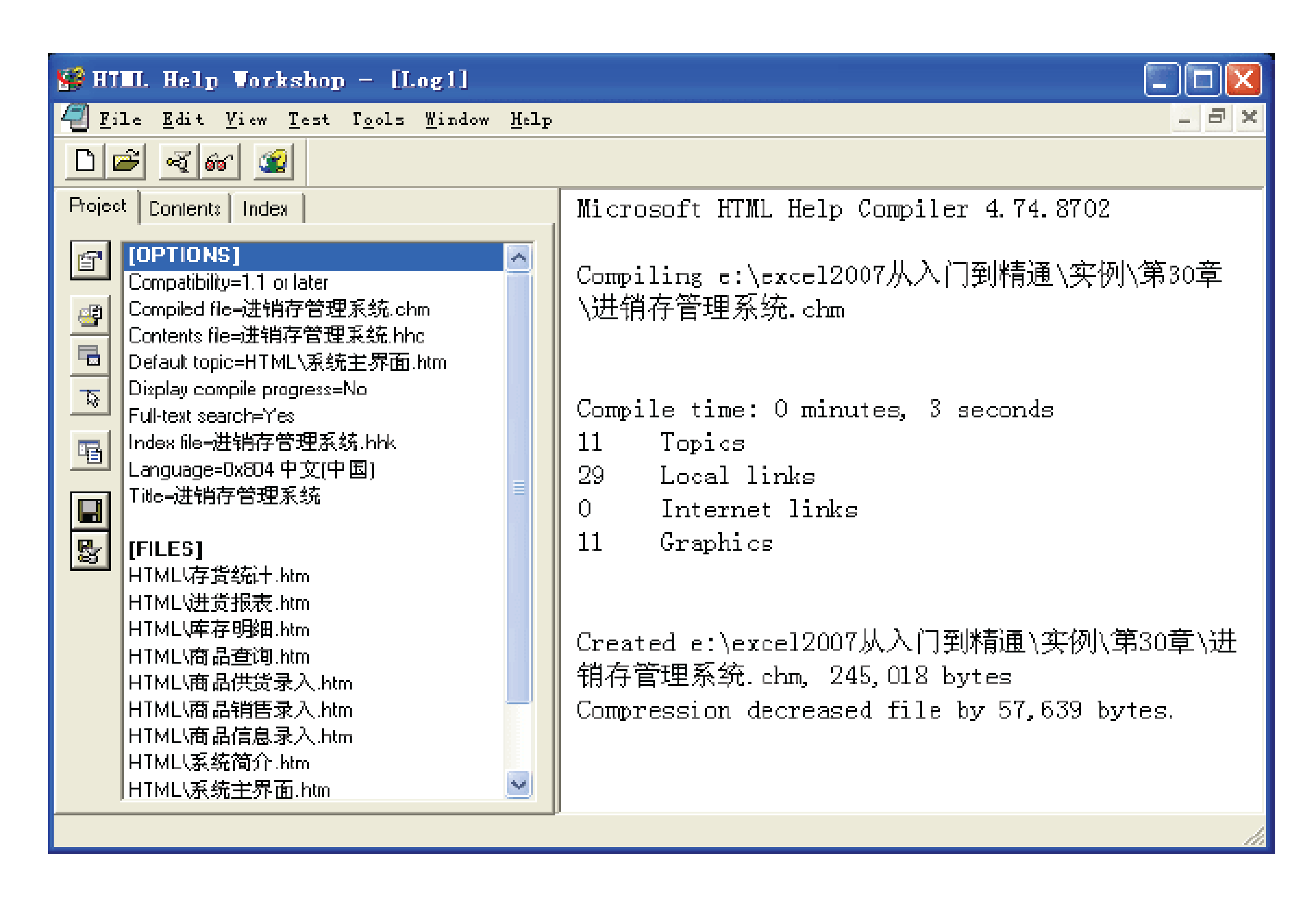


图 30-31　编译结果

30.3.6 打开帮助文件

编译生成的 CHM 文件可在 Windows 环境下直接打开使用，也可通过链接在应用程序中使用。

下面先在 HTML Help Workshop 中查看帮助文件是否达到需要的效果。

（1）单击菜单 View|Compiled file 命令，打开如图 30-32 所示对话框，单击其中的 Browse 按钮，找到需要查看的 CHM 文件。

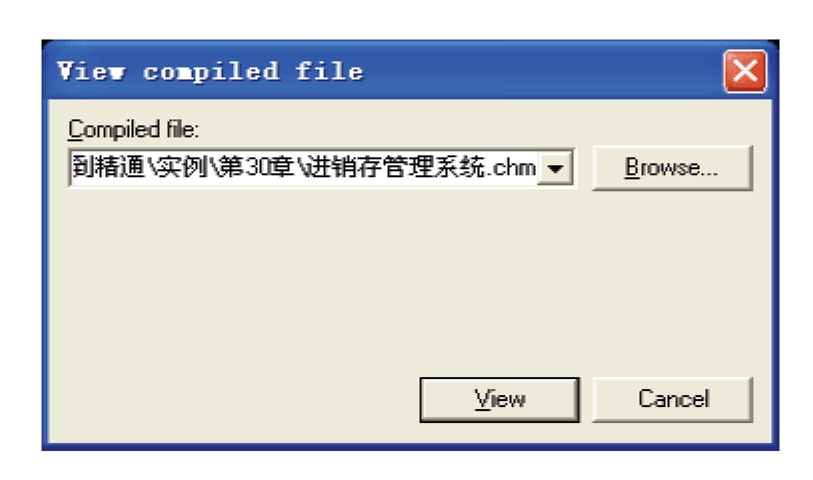


图 30-32　查看编译后的文件

（2）单击 View 按钮，即可看到帮助文件，如图 30-33 所示。

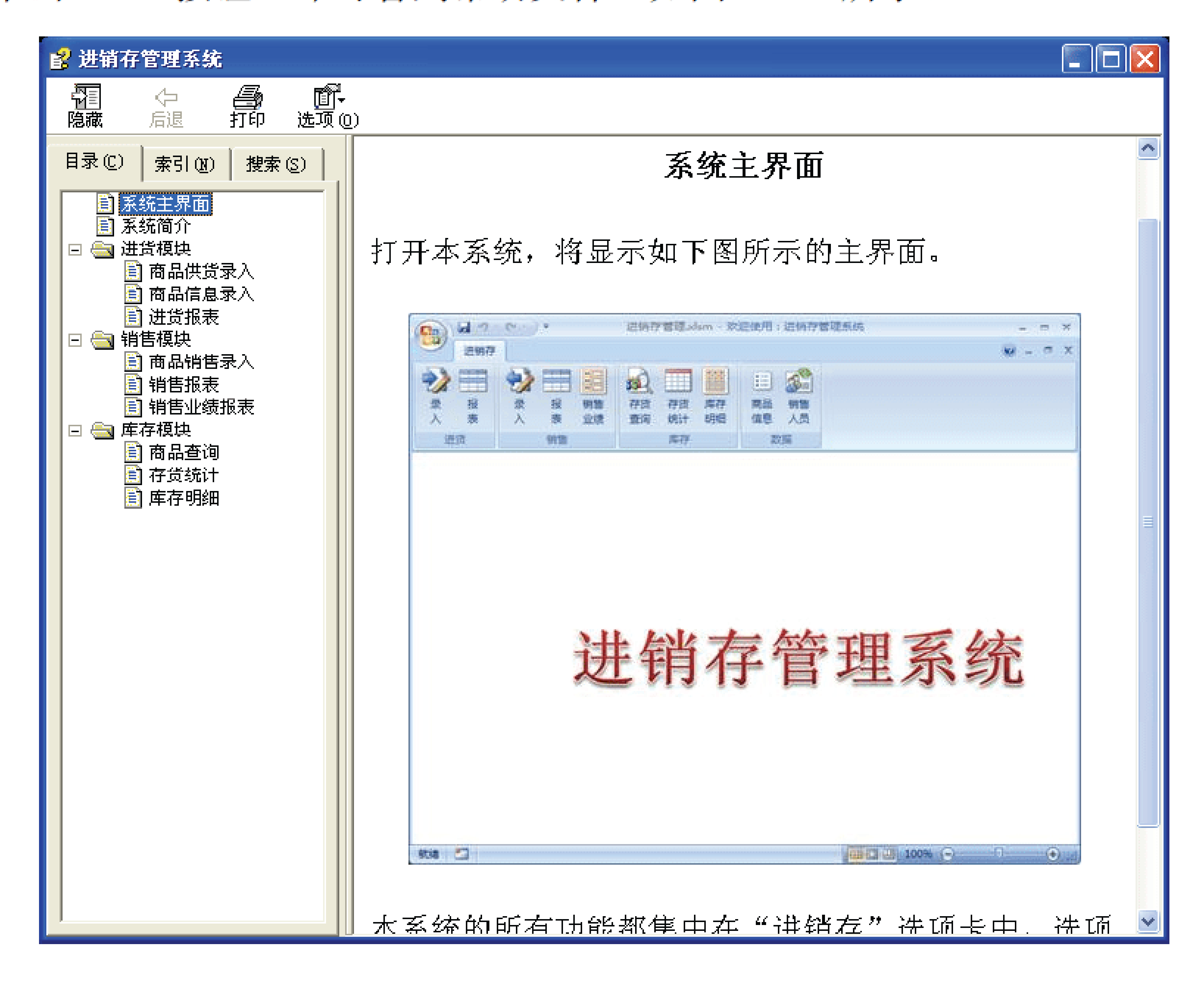


图 30-33　帮助文件主界面

（3）单击展开左侧的目录，可分别单击对应的主题查看。也可在【索引】选项卡中按关键字查看帮助主题。还可在【搜索】选项卡中可对帮助文件进行全文检索，其使用方法这里不再详细介绍。

30.4　给应用程序挂接帮助

通过 30.3.6 节的方法制作好一个 CHM 帮助文件后，还需要将该帮助文件与应用程序进行绑定，才能方便用户使用。

在 Windows 系统中，在大多数应用程序中按 F1 键将显示帮助。在 VBA 应用程序中按 F1 键时，将出现 Excel 的帮助系统。要显示自己制作的帮助系统，还需要编写一定的代码。

（1）将 CHM 帮助文件复制到与 Excel VBA 应用程序相同的一个文件夹下，以方便管理和程序的发布。

（2）在 VBA 应用程序的 Workbook 对象的 Open 事件中编写以下代码：

```
Private Sub Workbook_Open()
    Application.OnKey "{F1}", "CustHelp"
End Sub
```

其中，Application 对象的 OnKey 方法设置当按特定键或特定的组合键时运行指定的过程。这里指定当按 F1 键时，调用 CustHelp 过程。

（3）向工程中插入一个模块，或在原有模块中编写 CustHelp 过程的代码如下：

```
Sub CustHelp()
    Application.Help (ThisWorkbook.Path & "\进销存管理系统.chm")
End Sub
```

其中使用 Application 对象的 Help 方法来显示一个帮助主题，帮助文件作为方法的参数。

通过以上步骤的设置，在应用程序中按 F1 键时就可打开上节制作的帮助文件。

还可以在应用程序中设置按钮、快捷键、选项卡等功能键来调用 CustHelp 过程，从而为应用程序提供多方面的帮助。

第 7 部分　综合应用程序设计

本书前面用了 30 章的篇幅，详细介绍了 Excel VBA 开发时需要用到的相关知识。本部分介绍在 Excel 2007 中开发进销存管理系统的过程，以进一步巩固前面所学的知识。

第 31 章　进销存管理系统

进销存管理系统广泛应用于商场、超市、购物中心等零售行业，可对商品的购入、销售、库存进行有效的跟踪管理。本章将介绍用 Excel 制作小型的进销存管理系统，可用于小型企业对商品进行管理。

31.1　系 统 描 述

本例制作一个简单的进销存管理系统，主要完成商品的采购、销售、库存管理，还需按销售人员统计销售业绩。

进销存管理系统中将设计如图 31-1 所示的模块。

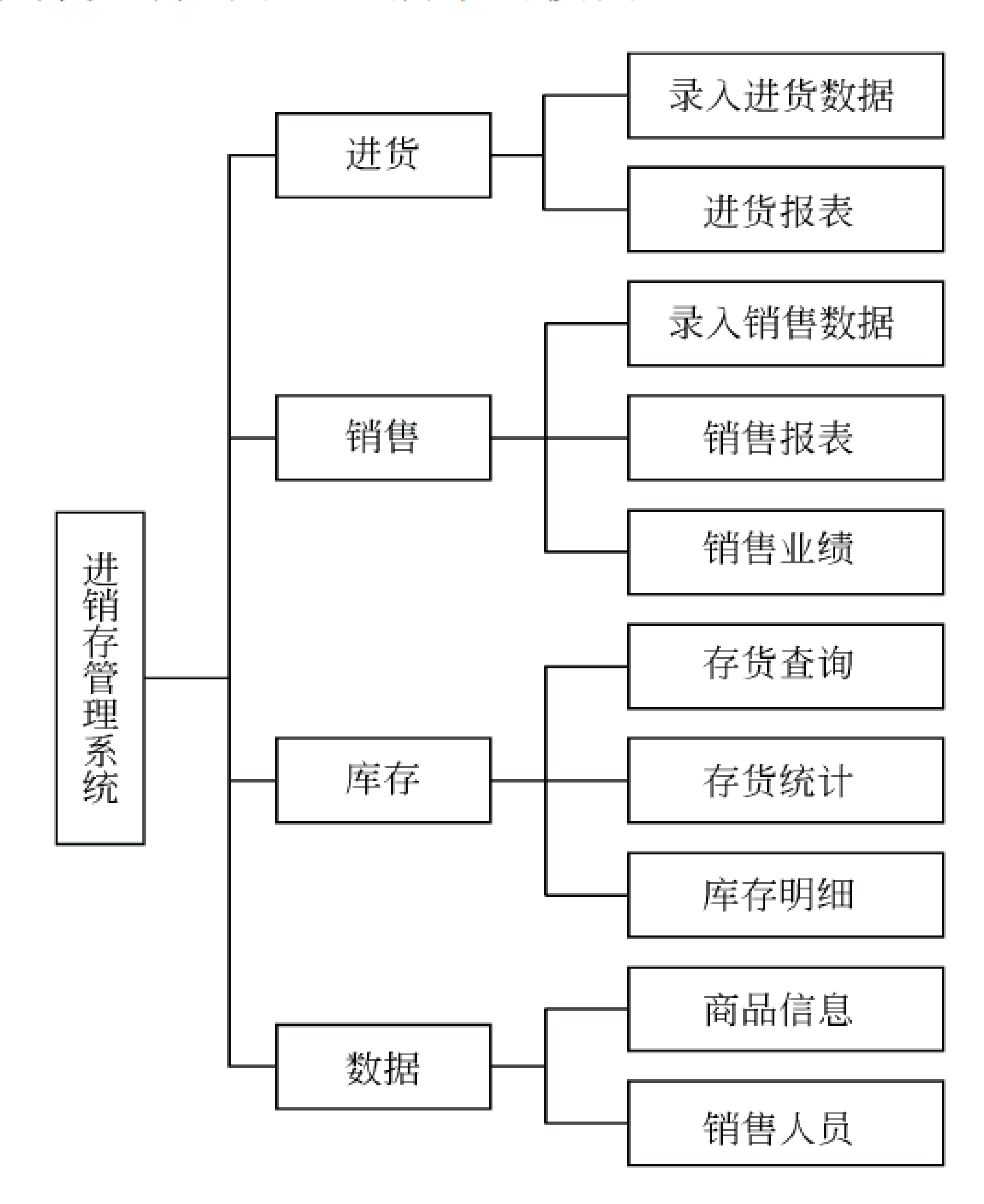


图 31-1　系统功能模块

1. 进货

进货模块包括下面两个功能。

- 录入进货数据：可录入进货数据。在录入进货数据时，还可根据需要增加商品信息。

- ❑ 进货报表：浏览指定期间的进货情况。

2．销售

销售模块包括下面三个功能。

- ❑ 录入销售数据：录入商品货号后，系统自动从商品信息中提取相关数据，然后输入销售数量，如果销售价格与设定的零售价格有差别，可输入到“调整价”中。
- ❑ 销售报表：可按设置的日期期间统计销售数据。
- ❑ 销售业绩：按指定日期期间统计销售员的业绩。

3．库存

库存模块包括下面三个功能。

- ❑ 存货查询：按货号查询商品的购进、销售、库存情况。
- ❑ 存货统计：统计所有库存商品的信息。
- ❑ 库存明细：按日期期间统计指定货号的进、销、存明细数据。

4．数据

数据模块包括下面两个功能。

- ❑ 商品信息：录入或修改商品的信息（例如货号、名称、型号、各种价格等）。
- ❑ 销售人员：对销售人员进行管理。

31.2　表 格 设 计

本系统使用 Excel 工作表保存数据，本节将列出一些简单工作表的名称及表格结构，对于复杂的表格，将在介绍各模块具体功能时给出。表格中的数据将由 VBA 程序自动填充。

31.2.1　主界面

工作表“主界面”不做具体的操作，只是显示一个文字提示。将工作表的名称修改为“主界面”，向工作表中插入艺术字“进销存管理系统”，如图 31-2 所示。

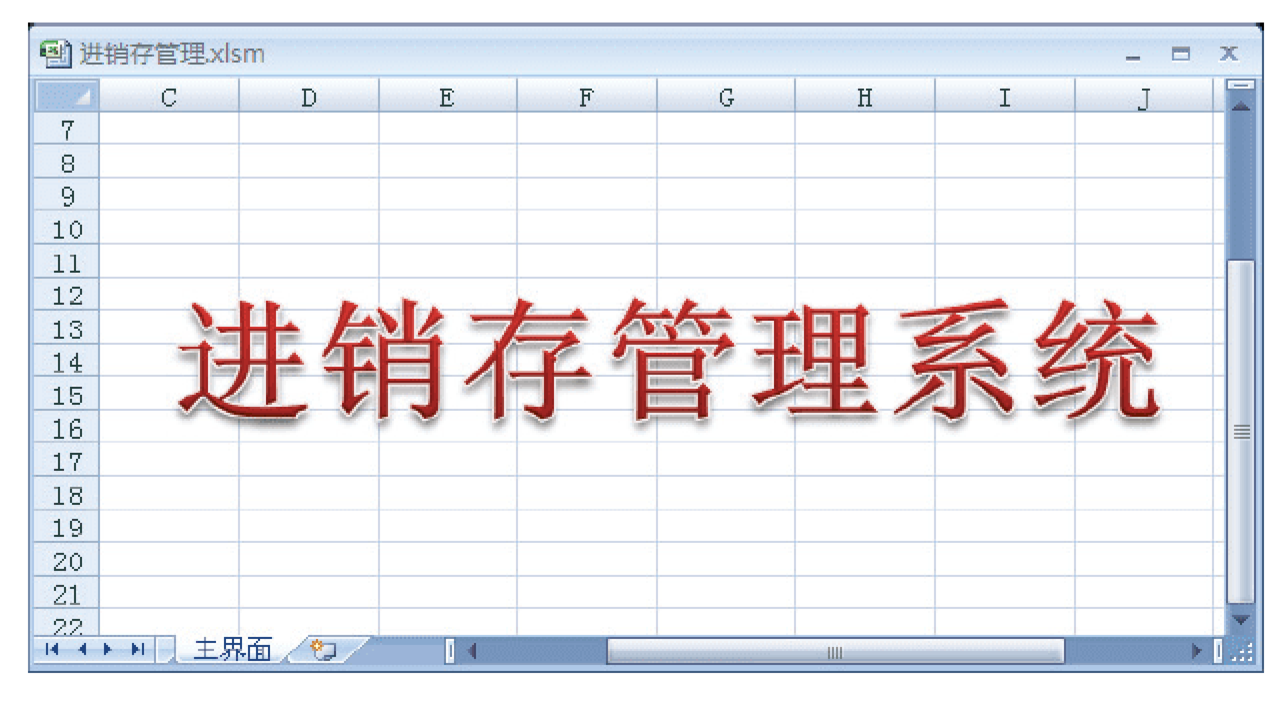


图 31-2　主界面

为了防止用户单击选择工作表中的单元格和艺术字，在工作表的双击事件（BeforeDoubleClick）和右击事件（BeforeRightClick）中编写代码，禁止响应用户的这两个鼠标操作，具体代码如下：

```
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As
Boolean)
    Cancel = True
End Sub
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range, Cancel As
Boolean)
    Cancel = True
End Sub
```

接着在【审阅】选项卡的【更改】组中，单击【保护工作表】按钮，在弹出的对话框中输入保护密码。将工作表保护以后，用户将不能选择工作表中的单元格。

31.2.2 商品信息

工作表“商品信息”用来保存系统中所有商品的信息，用户可以在该工作表中录入商品信息，也可在输入商品入库信息时输入商品信息。该工作表如图 31-3 所示。

货号	名称	型号	产地	进价	批价	零售	备注
1-111	长虹	29寸	四川	1800	1850	1900	
1-113	长虹空调	1.5P	四川	1500	1600	1800	
1-112	长虹	34寸	四川	2400	2600	2800	
1-121	长虹冰箱	190升	四川	1500	1600	1700	
2-111	海尔冰箱	180升	青岛	1400	1500	1700	
2-100	海尔空调	1.5P	青岛	1850	1950	2100	

图 31-3　商品信息

31.2.3 销货

工作表“销货”用于保存销售商品的所有信息，用户在“销货单”中录入的数据将保存在该工作表中，如图 31-4 所示。该表中的数据由程序自动填充，用户不能手工输入数据。



日期	货号	名称	型号	数量	单价	金额	调整价	备注	购货人	销售人
2008-4-18 15:54	1-111	长虹	29寸	1	1900	1900				李四
2008-4-18 15:54	2-100	海尔空调	1.5P	2	2100	4200				李四

图 31-4 销货

31.2.4　供货

工作表“供货”保存采购的商品信息，用户在“供货单”中录入的数据将保存在该工作表中，如图 31-5 所示。该表中的数据由程序自动填充，用户不能手工输入数据。

日期	货号	名称	型号	产地	进价	批价	零售	数量	金额	供应商
2008-4-17 15:51	1-112	长虹	34寸	四川	2400	2600	2800	5	12000	长丰公司
2008-4-17 15:51	1-113	长虹空调	1.5P	四川	1500	1600	1800	5	7500	长丰公司
2008-4-17 15:51	2-111	海尔冰箱	180升	青岛	1400	1500	1700	15	21000	长丰公司
2008-4-17 15:51	1-121	长虹冰箱	190升	四川	1500	1600	1700	10	15000	长丰公司
2008-4-17 15:51	2-100	海尔空调	1.5P	青岛	1850	1950	2100	10	18500	长丰公司
2008-4-18 15:54	1-111	长虹	29寸	四川	1800	1850	1900	10	18000	长丰公司

图 31-5　供货

31.2.5　存货统计

工作表“存货统计”保存所有存货商品的信息。该表的数据将跟随进货、销售的情况不断变化，每次查看前都需要重新生成该表，生成该表的 VBA 代码在后面给出。这里只给出其格式，如图 31-6 所示。

货号	名称	总进量	总出量	库存量	单价	金额	备注
1-111	长虹	10	1	9	1800	16200	
1-113	长虹空调	5		5	1500	7500	
1-112	长虹	5		5	2400	12000	
1-121	长虹冰箱	10		10	1500	15000	
2-111	海尔冰箱	15		15	1400	21000	
2-100	海尔空调	10	2	8	1850	14800	

图 31-6　存货统计

31.2.6　销售人员

工作表“销售人员”管理公司所有销售人员，用户可在该工作表中添加、编辑销售人员。该表的结构如图 31-7 所示。



姓名	性别	年龄	电话	住址
张三				
李四				
王五				

图 31-7　销售人员

在录入销售数据时，将调用销售人员名称。因此，需要在该表中定义一个名称，该名称包括“销售人员”工作表中的姓名列的内容。在【公式】选项卡的【定义的名称】组中，

单击【定义名称】按钮，打开如图 31-8 所示的【新建名称】对话框。

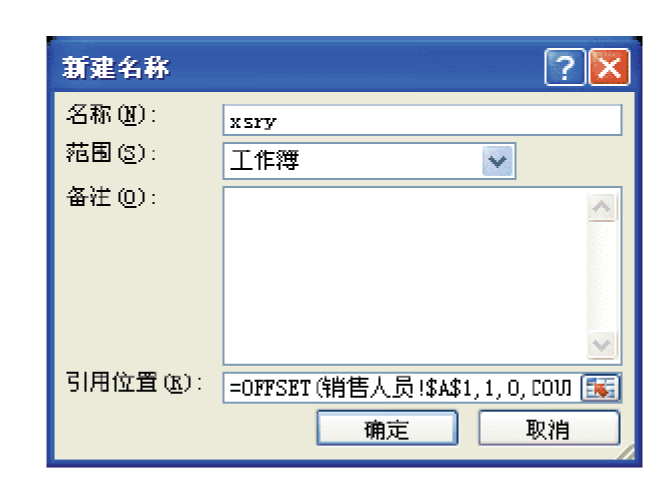


图 31-8 【新建名称】对话框

在【名称】文本框中输入 xsry，在【引用位置】文本框中输入以下公式：

```
=OFFSET(销售人员!$A$1,1,0,COUNTA(销售人员!$A:$A)-1,1)
```

以上公式根据工作表中的数据动态定义名称所包含的数据。

提示： 在本系统最后完成时，应将各工作表通过密码保护起来，密码设为 wyg，不知道密码的普通用户将不能修改数据（不建议用户在表格中修改数据）。表格中各项数据的列数不要随意调整，因为在 VBA 代码中将按对应单元格进行调用。

31.3 设计功能区

本系统使用 Excel 2007 进行开发，所以不需设置菜单和工具栏，调用各模块的功能都集中到功能区中。本节主要介绍设计功能区的方法。

31.3.1 设计功能区的 XML

按 31.2.6 节的格式制作各工作表，将工作表保存为“进销存管理.xlsm”。接着按以下步骤设置功能区的 XML。

（1）在当前文件夹中创建一个名为 customUI 的文件夹。

（2）打开【记事本】程序，输入以下内容：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
<commands>
    <command idMso="FileNew" enabled="false" />
    <command idMso="FileSaveAs" enabled="false" />
    <command idMso="FileSaveAsMenu" enabled = "false" />
    <command idMso="FileOpen" enabled="false" />
    <command idMso="FilePrintQuick" enabled="false" />
    <command idMso="FileClose" onAction="MyClose" />
    <command idMso="MergeCenterMenu" enabled = "false" />
</commands>
  <ribbon>
```

```
  <tabs>
    <tab idMso="TabHome" visible="false"/>
    <tab idMso="TabInsert" visible="false"/>
    <tab idMso="TabPageLayoutExcel" visible="false"/>
    <tab idMso="TabFormulas" visible="false"/>
    <tab idMso="TabData" visible="false"/>
    <tab idMso="TabReview" visible="false"/>
    <tab idMso="TabView" visible="false"/>
    <tab idMso="TabDeveloper" visible="false"/>
    <tab idMso="TabAddIns" visible="false"/>
    <tab id="tabJXC"   label="进销存" >
        <group id="gpIN" label="进货">
            <button id="btIN_INPUT" imageMso="ReviewNextChange"
                  size="large"  label="录入" onAction="btIN_INPUT_onAction"/>
            <button id="btIN_REPROT" imageMso="RecordsMoreRecordsMenu"
                  size="large"  label="报表" onAction="btIN_REPORT_
                  onAction"/>
         </group>
        <group id="gpOUT" label="销售">
            <button id="btOUT_INPUT" imageMso="ReviewPreviousChange"
                  size="large"  label="录入" onAction="btOUT_INPUT_
                  onAction"/>
            <button id="btOUT_REPROT" imageMso="RecordsMoreRecordsMenu"
                  size="large"  label="报表" onAction="btOUT_REPORT_
                  onAction"/>
            <button id="btOUT_RESULTS" imageMso="ControlLayoutStacked"
                  size="large"  label="销售业绩" onAction="btOUT_RESULTS_
                  onAction"/>
         </group>
        <group id="gpSTOCK" label="库存">
            <button id="btSTOCK_FIND" imageMso="FileDocumentManagement-
            Information"
                  size="large"  label="存货查询" onAction="btSTOCK_FIND_
                  onAction"/>
            <button id="btSTOCK_SUM" imageMso="AccessFormDatasheet"
                  size="large"  label="存货统计" onAction="btSTOCK_SUM_
                  onAction"/>
            <button id="btSTOCK_DETAILS" imageMso="ControlLayoutTabular"
                  size="large"  label="库存明细" onAction="btSTOCK_DETAILS_
                  onAction"/>
         </group>
        <group id="gpDATA" label="数据">
            <button id="btDATA_COMM" imageMso="SmartArtAddBullet"
                  size="large"  label="商品信息" onAction="btDATA_COMM_
                  onAction"/>
            <button id="btDATA_SALES" imageMso="DistributionListUpdateMembers"
                  size="large"  label="销售人员" onAction="btDATA_SALES_
                  onAction"/>
         </group>
    </tab>
 </tabs>
```

```
  </ribbon>
</customUI>
```

以上 XML 代码首先禁止【Office 按钮】中的部分菜单功能，接着隐藏功能区 9 个内置选项卡，最后创建一个名为“进销存”的选项卡。

注意：因为 XML 要区分大小写，所以一定要注意字母的大小写。

（3）执行菜单【文件】|【保存】命令，在【另存为】对话框中的【保存类型】下拉列表框中选择“所有文件”，在【编码】下拉列表框中选择 UTF-8，将文件保存到当前文件夹的 customUI 文件夹下，名称为 customUI.xml。

（4）将工作簿文件“进销存管理.xlsm”重命名为“进销存管理.xlsm.zip”，使用 Excel 工作簿变为一个压缩文件。

（5）双击压缩文件，并用 WinRar 打开该文件，拖动当前文件夹下的 customUI 文件夹至打开的压缩文件窗口，如图 31-9 所示。

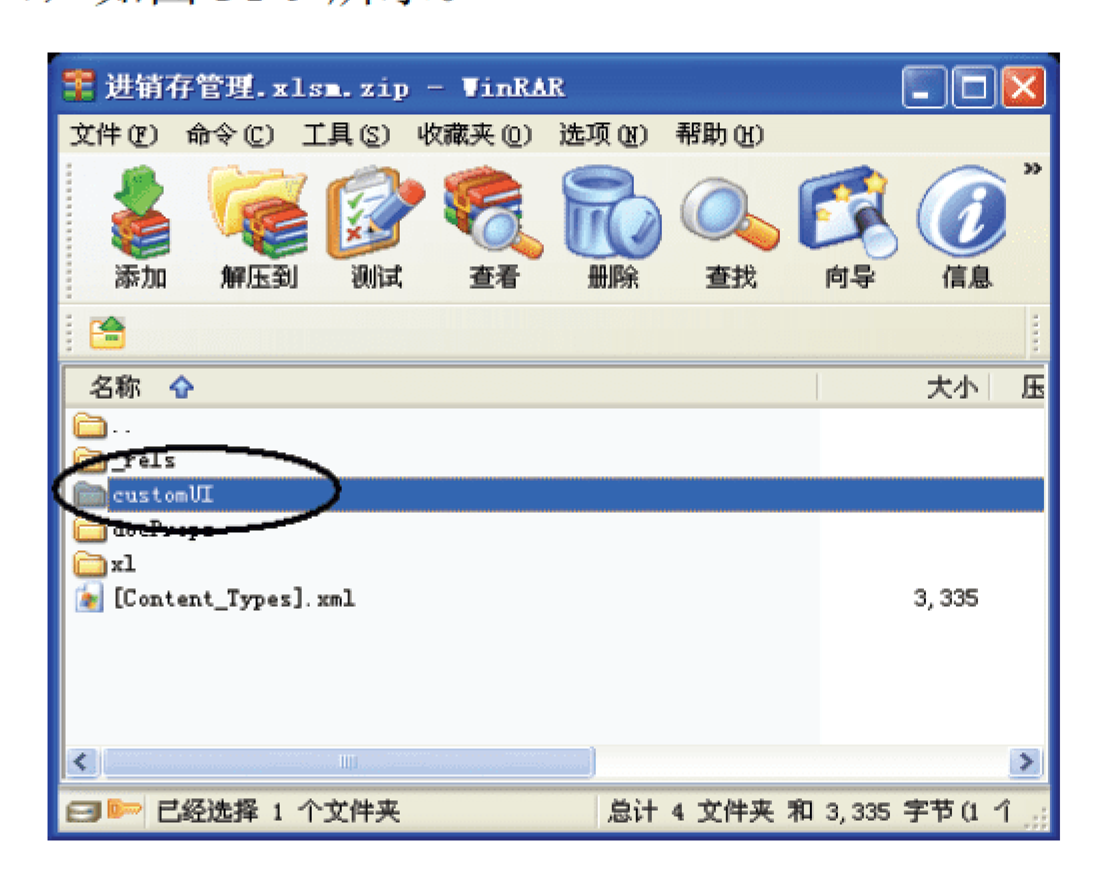


图 31-9　将 XML 添加到压缩文件

（6）将图 31-9 所示对话框中的_rels 文件夹拖到当前文件夹中。

（7）打开_rels 文件夹中的.rels 文件，在最后一个 Relationship 标记与 Relationships 标记之间添加以下内容，将工作簿文件与 customUI 文件夹中的 customUI.xml 文件之间创建关系。

```
<Relationship Id="customUIRelID"
Type="http://schemas.microsoft.com/office/2006/relationships/ui/extensibility" Target="customUI/customUI.xml" />
```

（8）保存并关闭.rels 文件。

（9）删除图 31-9 所示对话框中的_rels 文件夹，然后拖动第 7 步修改_rels 文件夹到压缩文件窗口中。

（10）关闭压缩文件窗口，将压缩文件“进销存管理.xlsm.zip”重命名为“进销存管理.xlsm”。

（11）在 Excel 2007 中打开“进销存管理.xlsm”工作簿，将看到如图 31-10 所示自定

义选项卡。



图 31-10　自定义的功能区

31.3.2　设计功能区各按钮代码

设计制作好如图 31-10 所示功能区的选项卡后，还需要在 VBE 中编写 VBA 代码，以响应各按钮的单击事件。具体步骤如下：

（1）按组合键 Alt+F11 打开 VBE 环境。

（2）执行主菜单【插入】|【模块】命令增加一个标准模块。

（3）在【模块 1】代码窗口中输入以下过程代码，定义一个全局变量：

```
Option Explicit
Public Const AppName As String = "进销存管理系统"
```

（4）【进货】组中有两个按钮，用来操作进货录入和进货报表两个功能，具体的代码如下：

```
Sub btIN_INPUT_onAction(Control As IRibbonControl) '进货录入
    Sheets("供货单").Select                        '选择"供货单"工作表
    Range("b5").Select                             '选择 B5 单元格
End Sub
Sub btIN_REPORT_onAction(Control As IRibbonControl) '进货报表
    Sheets("进货报表").Select                      '选择"进货报表"工作表
    Range("C2").Select                             '选择 C2 单元格
End Sub
```

（5）【销售】组中有 3 个按钮，用来操作销售录入、销售报表和销售业绩三个功能，具体的代码如下：

```
Sub btOUT_INPUT_onAction(Control As IRibbonControl)  '销售录入
    Sheets("销货单").Select                          '选择"销货单"工作表
    Range("g9").Select                               '选择 G9 单元格
End Sub
Sub btOUT_REPORT_onAction(Control As IRibbonControl) '销售报表
    Sheets("销售报表").Select                        '选择"销售报表"工作表
    Range("C2").Select                               '选择 C2 单元格
End Sub
Sub btOUT_RESULTS_onAction(Control As IRibbonControl) '销售业绩
    Sheets("销售业绩报表").Select                    '选择"销售业绩报表"工作表
    Range("B2").Select                               '选择 B2 单元格
```

```
End Sub
```

（6）【库存】组中有 3 个按钮，用来操作存货查询、存货统计和库存明细三个功能，具体的代码如下：

```
Sub btSTOCK_FIND_onAction(Control As IRibbonControl)       '存货查询
    Sheets("商品查询").Select                              '选择"商品查询"工作表
    Range("d3").Select                                     '选择 D3 单元格
End Sub
Sub btSTOCK_SUM_onAction(Control As IRibbonControl)        '存货统计
    Sheets("存货统计").Select
End Sub
Sub btSTOCK_DETAILS_onAction(Control As IRibbonControl)    '库存明细
    Sheets("商品明细账").Select                            '选择"商品明细账"工作表
    Range("b2").Select                                     '选择 B2 单元格
End Sub
```

（7）【数据】组中有两个按钮，用来操作商品信息和销售人员两个功能，具体的代码如下：

```
Sub btDATA_COMM_onAction(Control As IRibbonControl)     '商品信息
    Sheets("商品信息").Select
    Range("A1").End(xlDown).Offset(1, 0).Select
End Sub
Sub btDATA_SALES_onAction(Control As IRibbonControl)    '销售人员
    Sheets("销售人员").Select
    Range("A1").End(xlDown).Offset(1, 0).Select
End Sub
```

31.4 进货模块

进货模块包括两个功能：录入进货数据和生成进货报表。在录入进货数据的同时，还可以录入商品信息。

31.4.1 商品供货录入

在进货时，使用工作表“供货单”录入进货数据，录入的数据保存到“供货”工作表中。该工作表的结构如图 31-11 所示。

在录入数据时，用户只需要输入货号，Excel 将自动参照“商品信息”工作表中的资料，将商品名称等信息填充到表格右侧各列中。若录入的货号在“商品信息”工作表中不存在，在单元格 B16 中将显示出来，单击【录入商品信息】按钮可录入商品信息。

下面介绍商品供货录入表格的设计及编写相应的代码。为了简化代码，在表格中能用 Excel 公式完成的功能尽量使用公式进行运算。

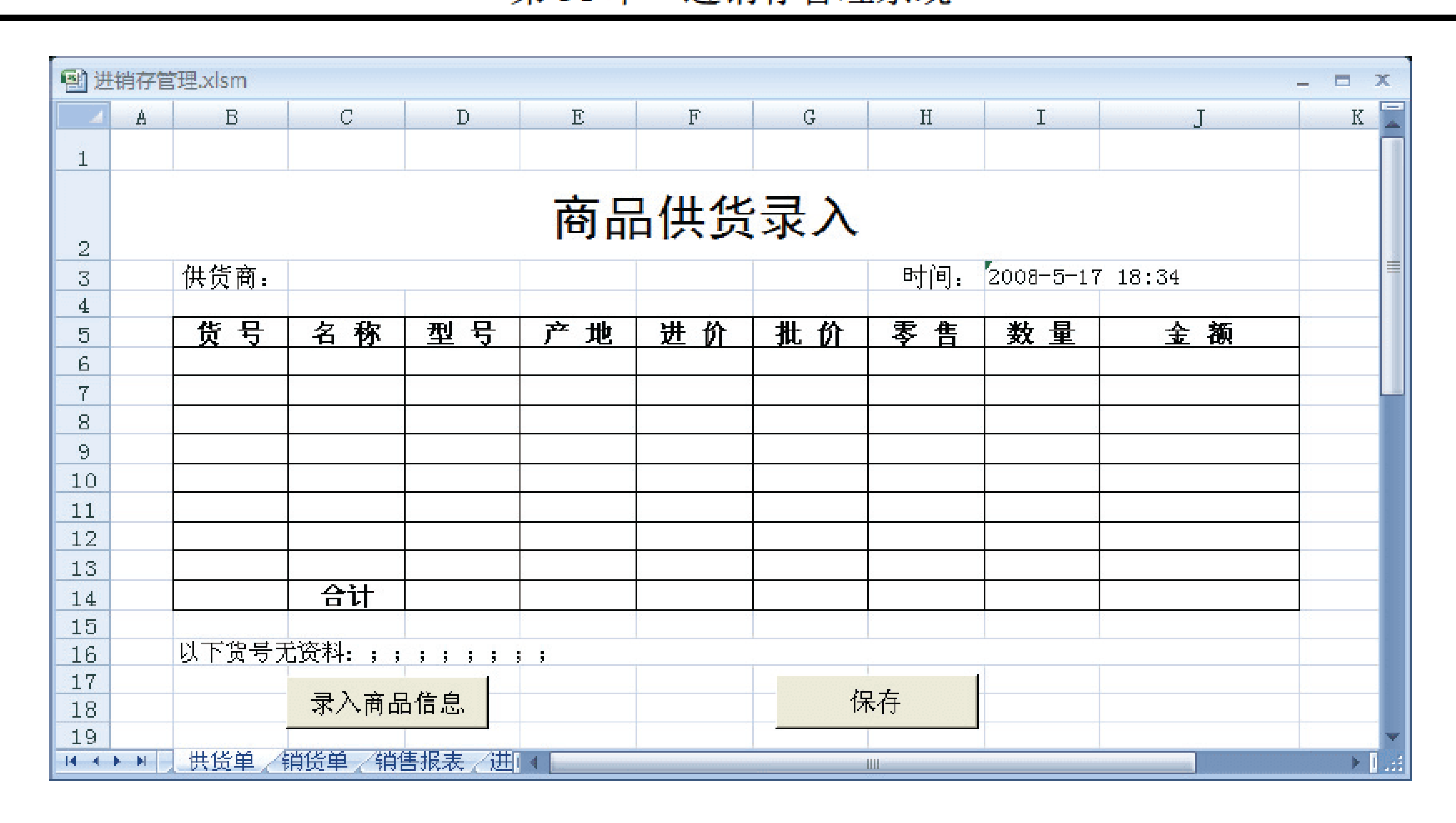


图 31-11　商品供货录入

1．设计表格

按图 31-11 所示结构制作工作表。

在录入采购信息时，首先录入供货商的名称，系统自动生成录入时间。然后逐行录入货号，系统根据货号自动从“商品信息”工作表中查找对应商品的信息，并显示在后面的单元格中。如果从“商品信息”工作表中没有查找到对应的商品信息，表格下方将显示需要录入的资料的货号。为了达到对应的功能，需在工作表对应的单元格设置公式。

（1）使单元格 I3 显示当前时间，设置公式如下：

```
=NOW()
```

（2）在 C6～H6 单元格和 J6 单元格分别设置以下公式，查表显示商品信息中的名称、型号、产地、进价、批价、零售和金额。

```
=IF($B6="","",VLOOKUP($B6,商品信息!$A:$H,2,FALSE))
=IF($B6="","",VLOOKUP($B6,商品信息!$A:$H,3,FALSE))
=IF($B6="","",VLOOKUP($B6,商品信息!$A:$H,4,FALSE))
=IF($B6="","",VLOOKUP($B6,商品信息!$A:$H,5,FALSE))
=IF($B6="","",VLOOKUP($B6,商品信息!$A:$H,6,FALSE))
=IF($B6="","",VLOOKUP($B6,商品信息!$A:$H,7,FALSE))
=IF(F6="","",F6*I6)
```

VLOOKUP 函数的功能是，在表格或数据值数组的首列查找指定的数值，并由此返回表格或数组当前行中其他列的值。其语法格式为：

```
VLOOKUP(lookup_value,table_array,col_index_num,range_lookup)
```

其中，各参数的含义如下面所述。

- ❑ lookup_value：为需要在表格或数组第一列中查找的数值。可以为数值或单元格引用。
- ❑ table_array：为两列或多列数据。请使用对区域的引用或区域名称。table_array 第

一列中的值是由 lookup_value 搜索的值。这些值可以是文本、数字或逻辑值，且不区分大小写。table_array 第一列中的值必须以升序排序，否则 VLOOKUP 可能无法返回正确的值。

- ❑ col_index_num：为 table_array 中待返回的匹配值的序列号。Col_index_num 为 1 时，返回 table_array 第一列中的数值；col_index_num 为 2，返回 table_array 第二列中的数值，以此类推。如果 col_index_num 小于 1，该函数将返回错误值#VALUE!，如果 col_index_num 大于 table_array 的列数，该函数将返回错误值#REF!。
- ❑ range_lookup：为逻辑值，指定希望 VLOOKUP 查找精确的匹配值还是近似匹配值。如果为 True 或省略，则返回精确匹配值或近似匹配值。也就是说，如果找不到精确匹配值，则返回小于 lookup_value 的最大数值。如果 Range_lookup 为 False，将返回精确匹配的值，如果找不到，则返回错误值#N/A。在此情况下，table_array 第一列的值不需要排序。如果 table_array 第一列中有两个或多个值与 lookup_value 匹配，则使用第一个找到的值。如果找不到精确匹配值，则返回错误值 #N/A。

（3）在工作表的 7～13 行分别复制上方的公式。

（4）设置 J14 单元格的公式如下，计算采购的总金额：

```
=SUM(J6:J13)
```

（5）在 B16 单元格设置以下公式，显示“商品信息”表中没有信息的货号。

```
="以下货号无资料："&IF(ISERROR(C6),B6,"")&";"&IF(ISERROR(C7),B7,"")&";
"&IF(ISERROR(C8),B8,"")&";"&IF(ISERROR(C9),B9,"")&";"&IF(ISERROR(C10),
B10,"")&"; "&IF(ISERROR(C11),B11,"")&";"&IF(ISERROR(C12),B12,"")&"; "&IF
(ISERROR(C13),B13,"")&"; "
```

（6）选中需要录入数据的单元格，将这些单元格的锁定取消。将工作表保护后，这些单元格仍然可以接收用户的输入。

2．设计代码

当用户输入好供货商品信息后，单击【保存】按钮即可将输入的数据保存到“供货”工作表中，需为该按钮编写宏代码。具体步骤如下：

（1）按组合键 Alt+F11 切换到 VBE 环境中。

（2）在【工程】资源管理器窗口中双击工作表“供货”打开代码窗口，编写以下子过程保存录入的数据：

```
Sub 保存供货信息()
    Dim x As Integer, i As Integer, j As Integer
    Call 手动计算
    With Sheets("供货")
        .Unprotect Password:="wyg"
        x = 2                           '从第 2 行开始
                                        '判断第 2 列的最后一行
        Do While Not (IsEmpty(.Cells(x, 2).Value))
            x = x + 1                   '在最后一行加一行即为空行
        Loop
    End With
```

```
    With Sheets("供货单")
        For i = 1 To 8
            If Not IsEmpty(.Cells(5 + i, 2)) Then
                Sheets("供货").Cells(x, 1) = .Cells(3, 9) '时间
                Sheets("供货").Cells(x, 1).NumberFormatLocal = "yyyy-m-d
                h:mm;@"
                Sheets("供货").Cells(x, 11) = .Cells(3, 3)  '供应商
                For j = 2 To 10
                    Sheets("供货").Cells(x, j) = .Cells(5 + i, j)
                Next j
                x = x + 1
            End If
        Next i
        .Range("b6:b13") = ""                          '清除供货单中的数据
        .Range("i6:i13") = ""
        .Select
    End With
    Call 自动计算
    With Sheets("供货").Range("A1").CurrentRegion
        .Borders.LineStyle = xlContinuous
        .Borders.Weight = xlThin
    End With
    Sheets("供货").Protect Password:="wyg"
End Sub
```

以上代码执行的流程如图 31-12 所示。

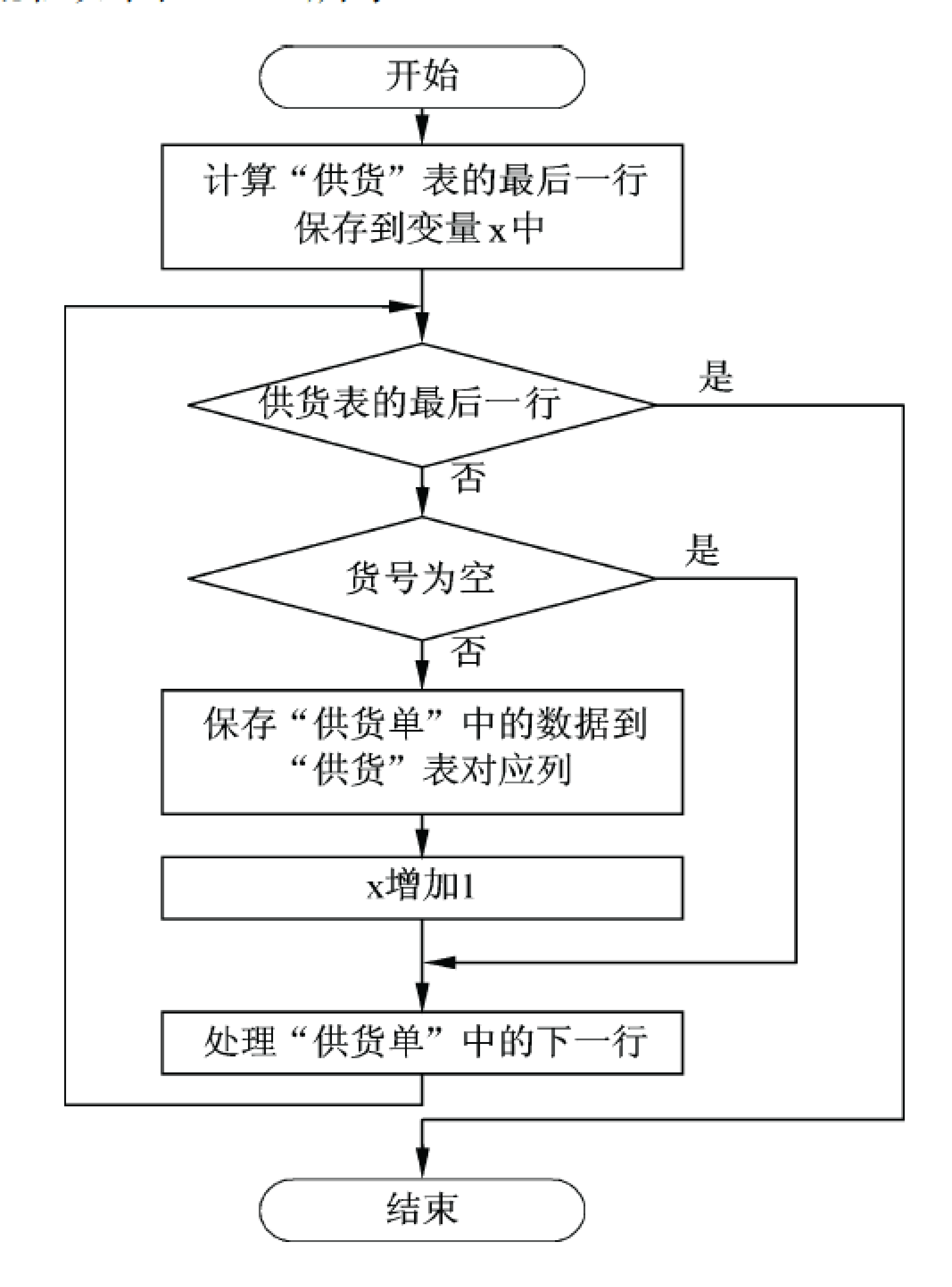


图 31-12　代码执行的流程图

（3）在上面的代码中调用了“手动计算”和“自动计算”子过程，这两个子过程用来设置 Excel 的计算模式，具体代码如下：

```
Sub 手动计算()
    With Application
        .Calculation = xlCalculationManual          '设置计算模式为手动
        .MaxChange = 0.001                          '设置迭代时的最大变化值
    End With
    ActiveWorkbook.PrecisionAsDisplayed = True
End Sub
Sub 自动计算()
    With Application
        .Calculation = xlCalculationAutomatic       '设置计算模式为自动
        .MaxChange = 0.001
    End With
    ActiveWorkbook.PrecisionAsDisplayed = True
End Sub
```

（4）当录入的货号在“商品信息”工作表中不存在时，单击【录入商品信息】按钮将跳转到“商品信息录入”表格，该按钮的宏代码如下：

```
Sub 录入商品信息()
    ActiveWindow.SmallScroll Down:=31                '工作表向下滚动 31 行
    Range("B38").Select                              '选择 B38 单元格
End Sub
```

其中 SmallScroll 方法用于滚动工作表到指定位置，由上向下滚动 31 行。

31.4.2　商品信息录入

在录入商品信息供货信息时，如果“商品信息”表中无货号对应的商品，则需要通过“商品信息录入”部分进行添加。

1．设计表格

商品信息录入表格如图 31-13 所示，该表格与“商品供货录入”放置在同一工作表中。

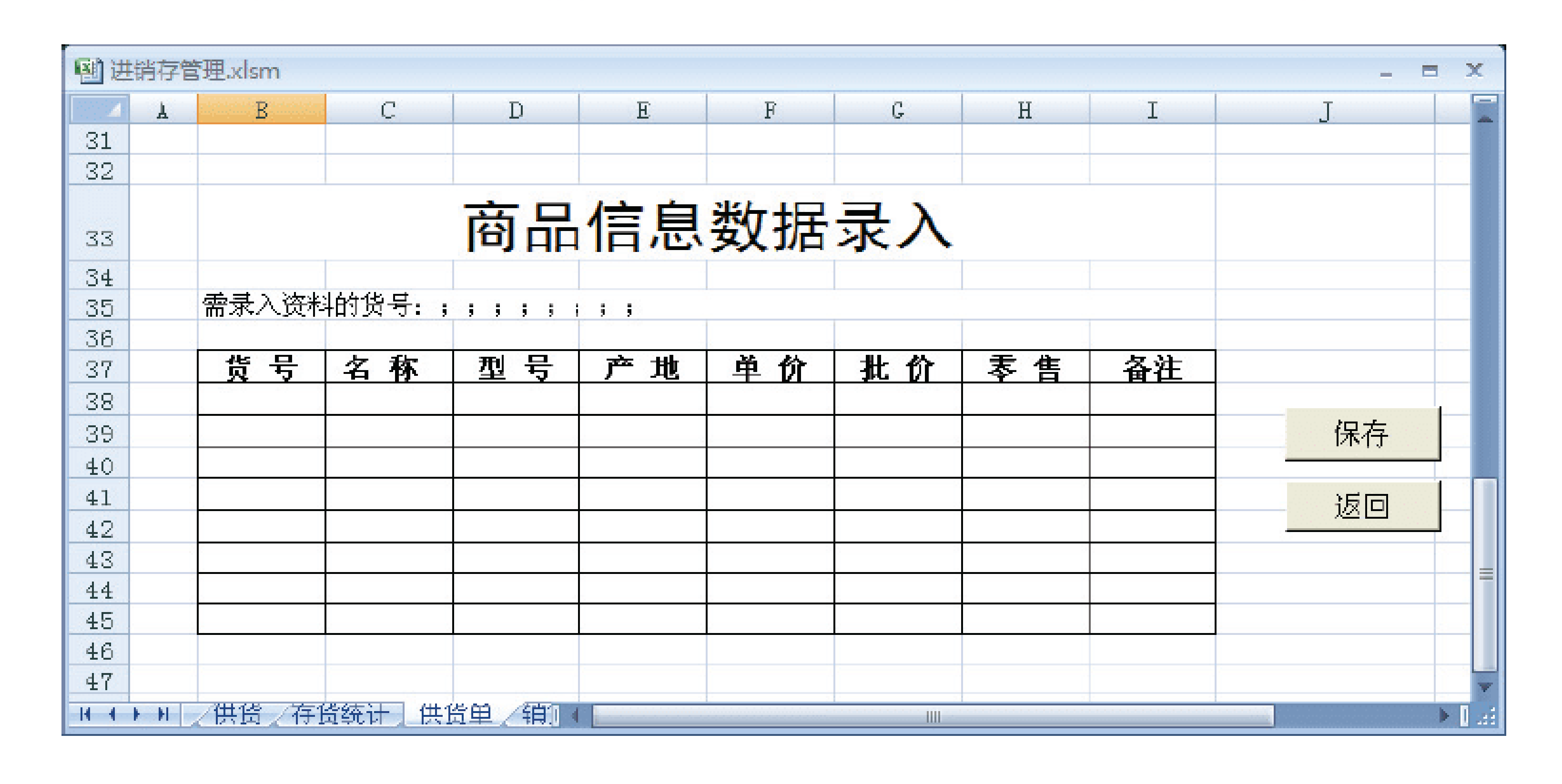


图 31-13　商品信息数据录入

2. 设置公式

商品信息录入的公式比较简单，只需要在单元格 B35 中设置以下公式，就可显示需要录入资料的货号：

```
="需录入资料的货号: "&IF(ISERROR(C6),B6,"")&"; "&IF(ISERROR(C7),B7,"")&";
"&IF(ISERROR(C8),B8,"")&";"&IF(ISERROR(C9),B9,"")&" ; "&IF(ISERROR(C10),
B10,"")&";"&IF(ISERROR(C11),B11,"")&";"&IF(ISERROR(C12), B12,"")&";"&IF
(ISERROR (C13),B13,"")&"; "
```

3. 设计代码

录入商品信息后，单击【保存】按钮，输入的数据将保存到“商品信息”工作表中，为该按钮编写的宏代码如下：

```
Sub 保存商品信息()
    Dim x As Integer, i As Integer, j As Integer
    Call 手动计算
    With Sheets("商品信息")
        x = 2                                    '从第 2 行开始
                                                 '判断第 2 列的最后一行
        Do While Not (IsEmpty(.Cells(x, 1).Value))
            x = x + 1                            '在最后一行加一行即为空行
        Loop
    End With
    Sheets("商品信息").Unprotect Password:="wyg"
    With Sheets("供货单")
        For i = 1 To 8
            If Not IsEmpty(.Cells(37 + i, 2)) Then
                For j = 1 To 8
                    Sheets("商品信息").Cells(x, j) = .Cells(37 + i, j + 1)
                Next j
                x = x + 1
            End If
        Next i
        .Range("B38:I45") = ""
        .Select
    End With
    Call 自动计算
    With Sheets("商品信息")
        .Range("A1").CurrentRegion.Select
        设置边框
        .Protect Password:="wyg"
    End With
End Sub
```

以上代码首先在“商品信息”工作表中查找最后的一个空行，接着将当前工作表中录入的数据填充到“商品信息”表的对应单元格，再调用“设置边框”子过程为“商品信息”工作表中所有的显示绘制边框。

“设置边框”子过程的代码很简单，调用其为选中的区域绘制外边框和内边框线，具体代码如下：

```
Sub 设置边框()
    Selection.Borders.LineStyle = xlContinuous
    Selection.Borders.Weight = xlThin
End Sub
```

用户单击【返回】按钮将回到“商品供货录入”表格，【返回】按钮的 VBA 代码如下：

```
Sub 返回供货录入()
    ActiveWindow.SmallScroll Down:=-31
End Sub
```

31.4.3　测试商品供货功能

完成模块的设计后，为了检验其正确性，可使用一些数据进行测试。具体步骤如下：

（1）在功能区【进销存】选项卡的【进货】组中，单击【录入】按钮，工作区域将显示如图 31-14 所示“商品供货录入”界面。

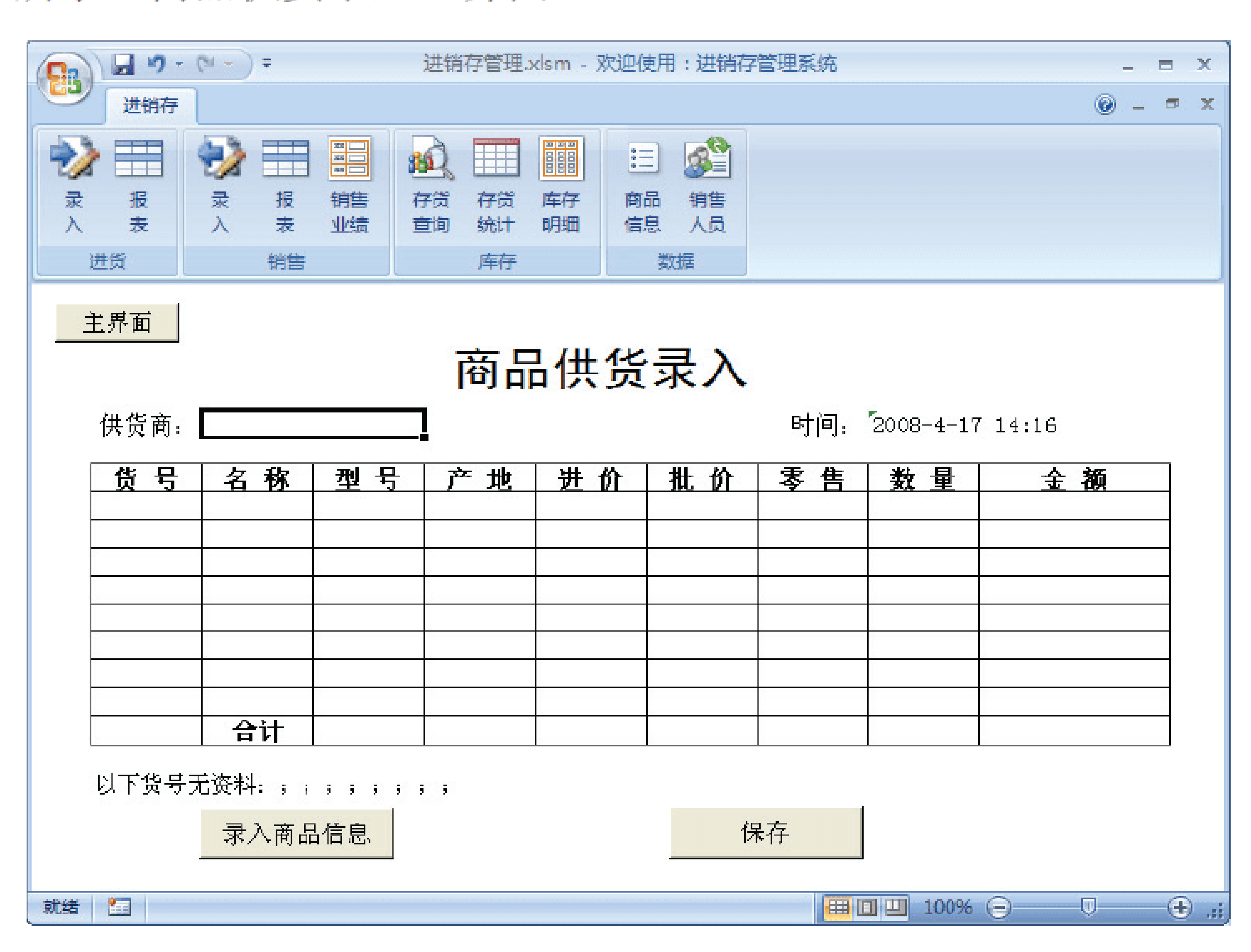


图 31-14　商品供货录入

在以上表格中，时间将自动填入，如果不正确，可修改计算机系统时间，再按快捷键 F9 进行刷新。

（2）在“供货商”后面输入相应的内容，然后输入一个货号“1-111”，系统自动查表填充相关数据，如图 31-15 所示。

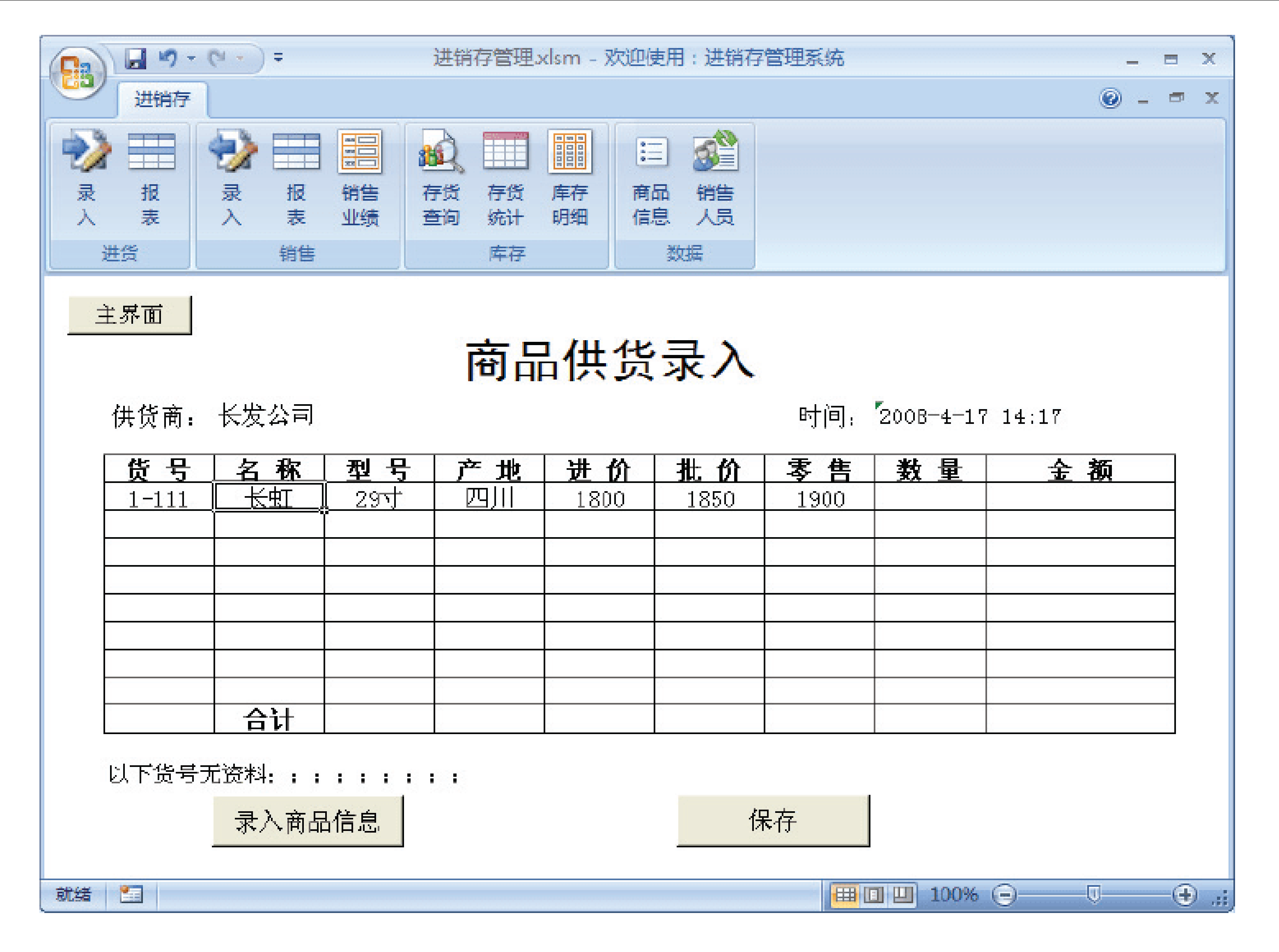


图 31-15　录入货号

（3）输入供货数量后，右侧的金额将自动生成。

（4）接着再输入一个货号“1-222”，因为“商品信息”表中没有该货号的资料，所以后面的名称等单元格显示为“#N/A”，表格下方也有提示，提醒用户货号“1-222”需要输入资料，如图 31-16 所示。

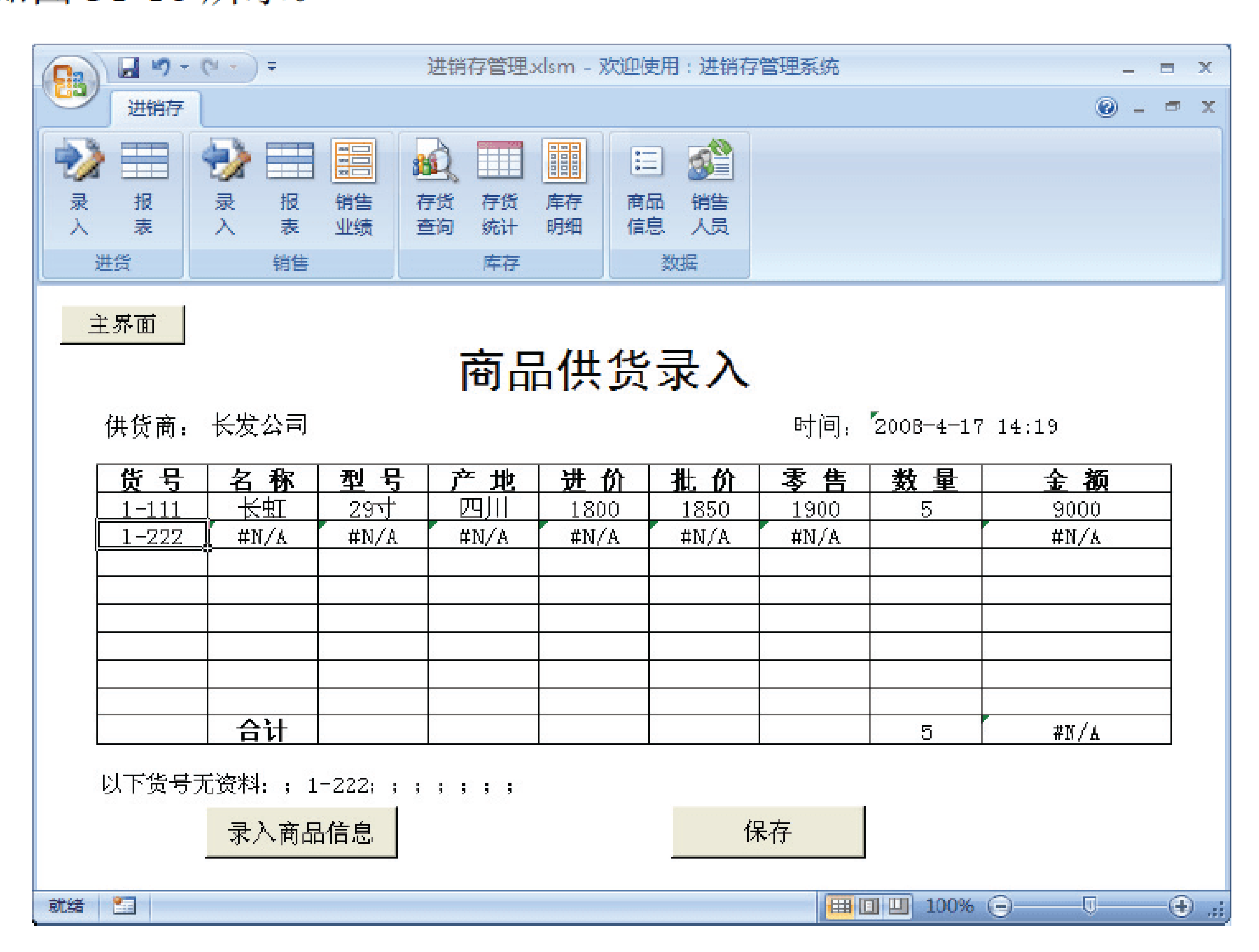


图 31-16　不存在的货号

（5）单击下方的【录入商品信息】按钮，滚动工作表到下方，录入商品的信息。“商品信息数据录入”的表格上方也显示有需要录入资料的货号，在表格中输入相关的信息，

可录入上面没有提示货号的商品信息，最后单击【保存】按钮将商品信息保存到对应的表格中，如图 31-17 所示。

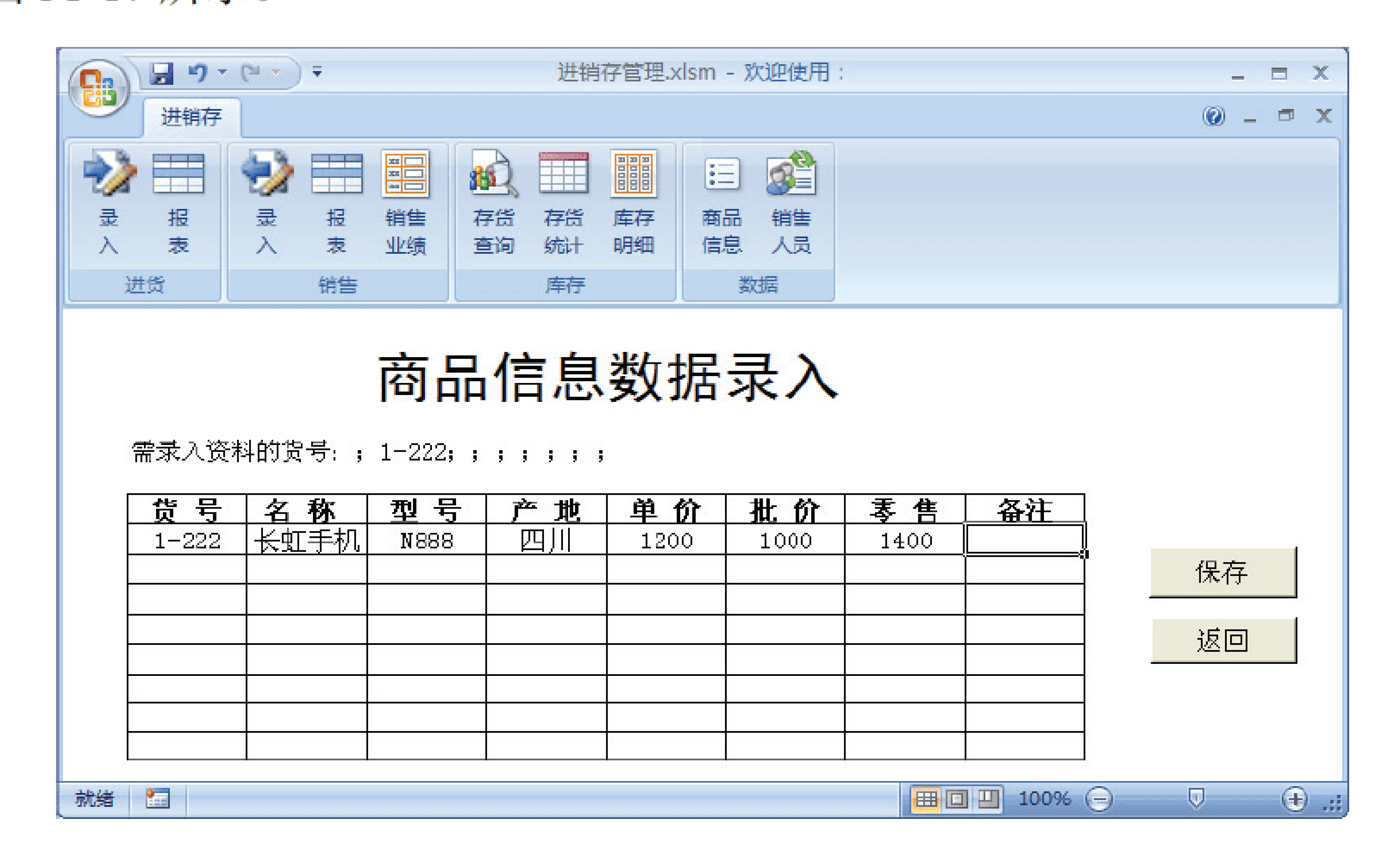


图 31-17　录入商品信息

（6）单击【返回】按钮，回到“商品供货录入”画面，可以看到，货号“1-222”对应的商品信息已显示完整，如图 31-18 所示。

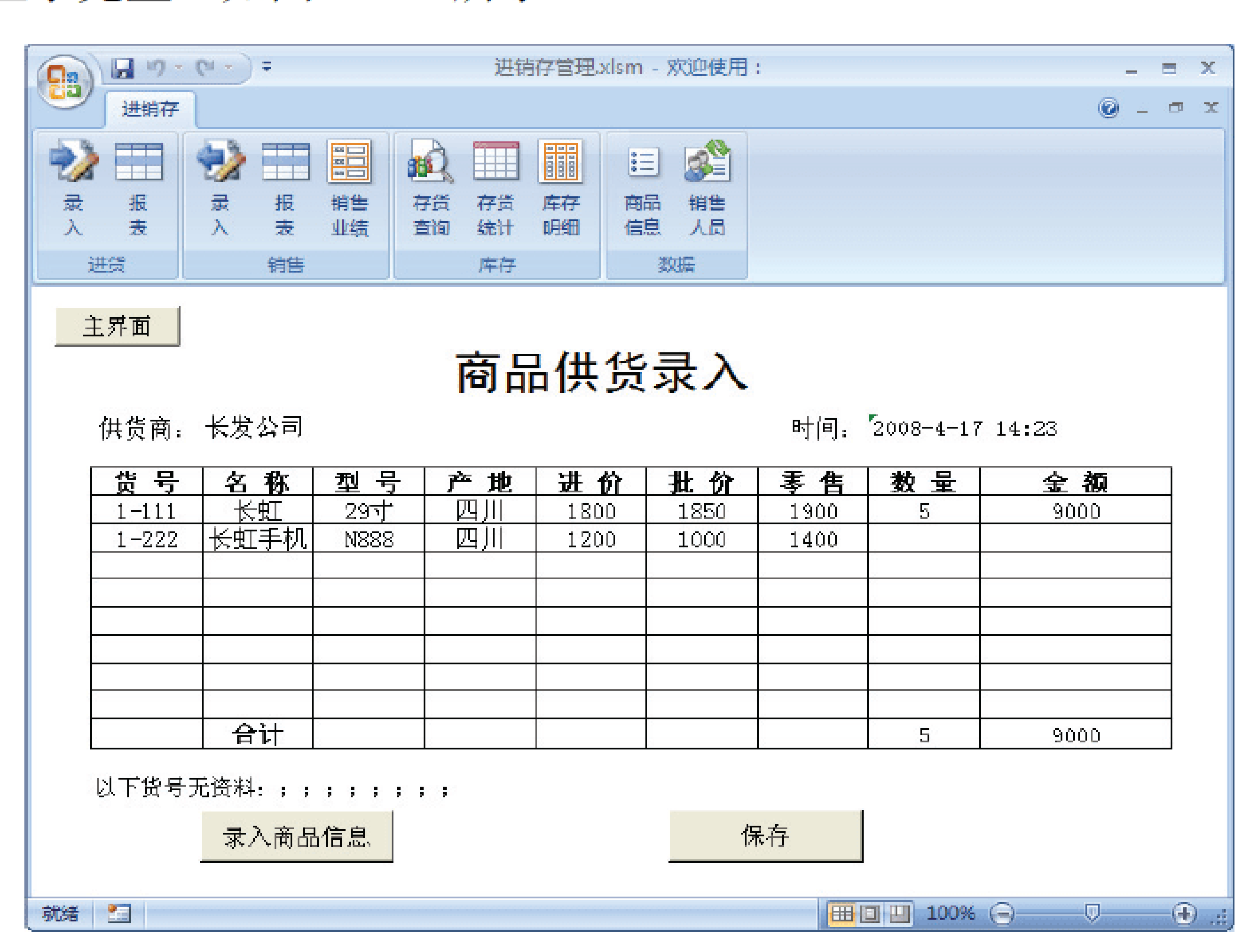


图 31-18　显示数据

（7）单击“商品供货录入”中的【保存】按钮，将供货信息保存到相应的工作表中后，即完成操作。如果查看“商品信息”工作表和“供货”工作表，可看到新添加的数据，如图 31-19 和图 31-20 所示。

提示：“供货”工作表的数据处于隐藏状态，用户不能查看。



图 31-19　新增商品信息



图 31-20　新增供货信息

31.4.4　进货报表

当用户在功能区【进销存】选项卡的【进货】组中，单击【报表】按钮时，将显示进货报表。在该报表中由用户输入统计的起止日期，程序将按时间段统计所有进货的情况，生成进货报表。

1．表格设计

首先制作如图 31-21 所示的表头，其中“日期”后面的两个单元格用来设置统计的起止日期，输入起止日期后，单击【生成报表】按钮，将生成指定时期内的进货详表。

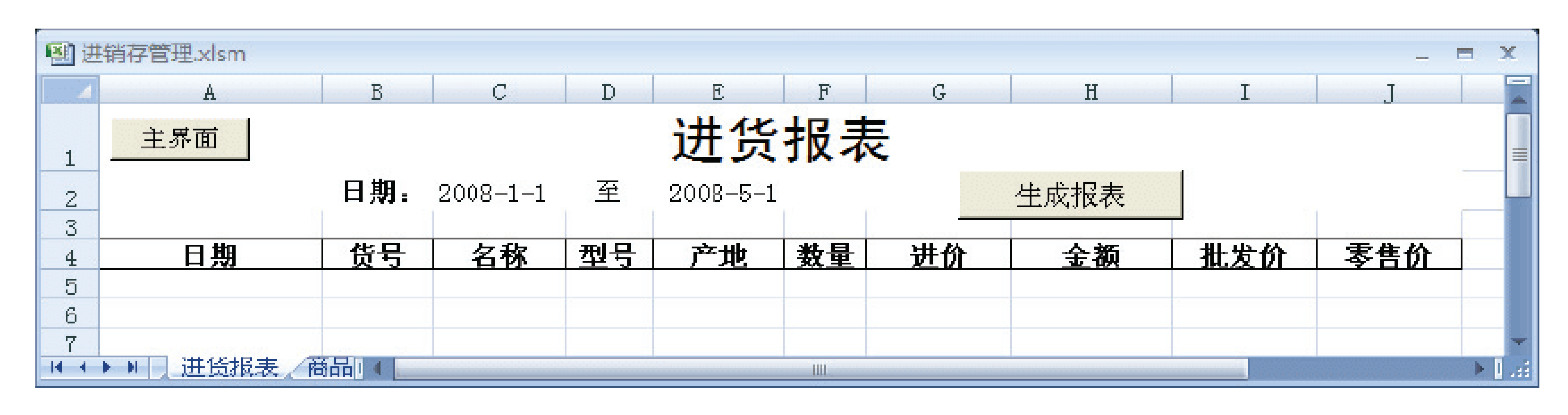


图 31-21　进货报表

2．编写代码

该报表主要通过 VBA 代码生成，编写代码的步骤如下。

（1）为【生成报表】按钮指定宏，并编写宏代码如下：

```
Sub 进货报表()
    Dim datStart As Date, datEnd As Date        '声明变量，保存起始日期值
    If IsDate(Cells(2, 3)) Then                 '起始日期格式正确
        datStart = DateValue(Cells(2, 3))       '保存到起始日期变量中
    Else                                        '起始日期格式错误
        MsgBox "日期输入错误"
        Exit Sub                                '退出子过程
    End If
    If IsDate(Cells(2, 5)) Then                 '结束日期格式正确
        datEnd = DateValue(Cells(2, 5))         '保存到结束日期变量中
    Else                                        '结束日期格式错误
        MsgBox "日期输入错误"
        Exit Sub                                '退出子过程
    End If
    If datStart > datEnd Then                   '起始日期大于结束日期
        MsgBox "起始日期应小于或等于结束日期"   '显示错误信息
    Else
        生成进货报表                            '调用子过程生成报表
    End If
End Sub
```

程序首先判断用户输入的日期格式，如果不正确，将弹出错误提示并退出子过程。接着判断起始日期是否小于或等于结束日期，最后调用“生成进货报表”子过程。

（2）检查日期正确后，将调用“生成进货报表”子过程生成进货报表，该过程流程图如图 31-22 所示。

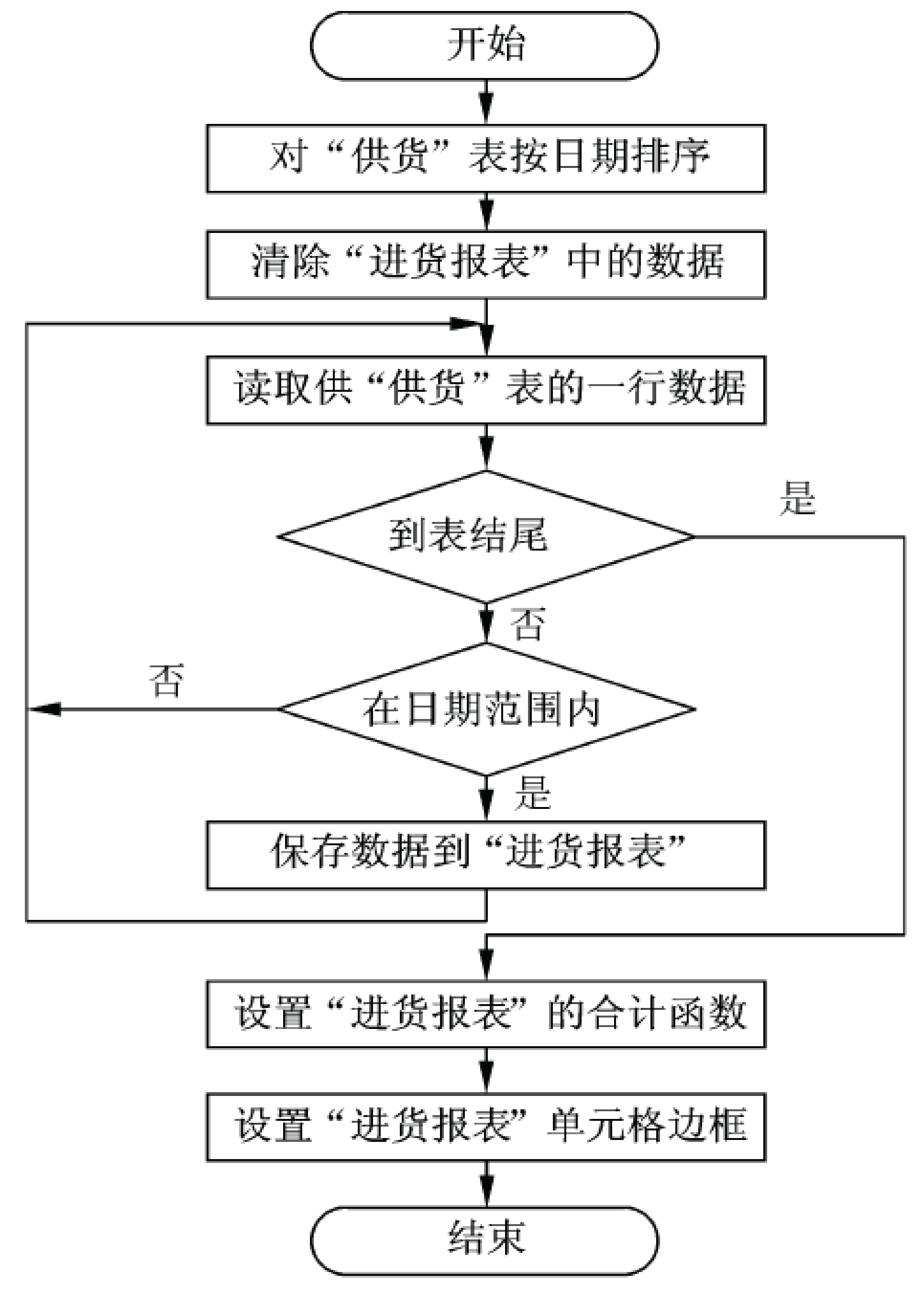


图 31-22　生成进货报表流程图

按图 31-22 所示流程图编写程序，代码如下：

```
Sub 生成进货报表()
    Dim x As Integer, j As Integer, rngTemp As Range    '声明变量
    Dim intRow As Integer
    Sheets("供货").Unprotect Password:="wyg"          '取消对“供货”工作表的保护
    Set rngTemp = Sheets("供货").Range("A2").CurrentRegion
                                                      '获取供货表中的有效数据区域
    rngTemp.Sort Key1:=Sheets("供货").Range("A2"), Order1:=xlAscending,
    Header:= _
        xlGuess, OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom, _
        SortMethod:=xlPinYin, DataOption1:=xlSortNormal
                                                      '供货表按货号排序
    Sheets("供货").Protect Password:="wyg"            '保护工作表
    With Sheets("进货报表")
    .Unprotect Password:="wyh"                        '取消对“进货报表”的保护
    intRow = .Range("A4").CurrentRegion.Rows.Count   '取得“进货报表”的数据行数
    .Range(.Cells(5, 1), .Cells(intRow + 4, 10)).Select    '选取“进货报表”
    的数据部分
    Selection.EntireRow.Delete                        '删除表体部分的数据
    .Range("A5").Select                               '选择 A5 单元格
    x = 2                                             '“供货”表从第 2 行开始处理
    j = 5                                              “进货报表”从第 5 行开始处理
    Do While Not (IsEmpty(Sheets("供货").Cells(x, 2).value))
        If DateValue(Sheets("供货").Cells(x, 1)) >= DateValue(.Cells(2, 3))
        And _
            DateValue(Sheets("供货").Cells(x, 1)) <= DateValue(.Cells(2, 5))
            Then
                                                 '供货时间大于等于设置的起始时间
                                                 '且小于等于设置的结束时间
            .Cells(j, 1).NumberFormatLocal = "yyyy-m-d h:mm;@"
                                                      '设置第 1 列的为日期格式
            .Cells(j, 1) = Sheets("供货").Cells(x, 1) '日期
            .Cells(j, 2) = Sheets("供货").Cells(x, 2) '货号
            .Cells(j, 3) = Sheets("供货").Cells(x, 3 '名称
            .Cells(j, 4) = Sheets("供货").Cells(x, 4) '型号
            .Cells(j, 5) = Sheets("供货").Cells(x, 5) '产地
            .Cells(j, 6) = Sheets("供货").Cells(x, 9) '数量
            .Cells(j, 7).NumberFormatLocal = "#,##0.00_ "
                                                      '设置进价列的显示格式
            .Cells(j, 7) = Sheets("供货").Cells(x, 6) '进价
            .Cells(j, 8).NumberFormatLocal = "#,##0.00_ "
                                                      '设置金额列的显示格式
            .Cells(j, 8) = .Cells(j, 6) * .Cells(j, 7) '金额
            .Cells(j, 9).NumberFormatLocal = "#,##0.00_ "
                                                      '设置批发价列的显示格式
            .Cells(j, 9) = Sheets("供货").Cells(j, 7) '批发价
            .Cells(j, 10).NumberFormatLocal = "#,##0.00_ "
```

```
                                                            '设置零售价列的显示格式
            .Cells(j, 10) = Sheets("供货").Cells(j, 8)   '零售价
            j = j + 1
        End If
        x = x + 1                                           '处理供货表中的下一行
    Loop
    .Cells(j, 1) = "合计"
    .Cells(j, 8).NumberFormatLocal = "#,##0.00_ "          '设置合计的显示格式
    .Cells(j, 8).FormulaR1C1 = "=SUM(R[" & 5 - j & "]C:R[-1]C)"
                                                            '设置合计的公式
    .Range(.Cells(5, 1), .Cells(j, 10)).Select             '选择“进货报表”的数据部分
    设置边框                                                '调用子过程为其设置边框线
    .Select                                                 '选择“进货报表”工作表
    .Range("A5").Select                                     '选择 A5 单元格
    .Protect Password:="wyg"                                '保护该工作表
    End With
End Sub
```

为了防止用户随意修改工作表，设计完成后，对每个工作表都设置了保护。在程序中对工作表进行操作之前，需先进行撤销保护操作，操作完成后，再进行保护操作。进行撤销保护操作时使用工作表的 Unprotect 方法，保护操作使用工作表的 Protect 方法。

在“进货报表”中输入日期值，然后单击【生成报表】按钮，得到如图 31-23 所示的进货报表。



图 31-23　生成进货报表

31.5　销售模块

销售模块用来管理销售相关的数据，包括录入销售数据、查看销售报表、查看销售业绩等功能。

31.5.1　设计销货单

销货单需要登记购买人，及其购买的商品和数量，为了考核员工的销售业绩，还需要登记销售人员。

1. 表格设计

表格如图 31-24 所示。“销售人员”设计为一个下拉列表框，用户可直接选择销售人员。要达到这种效果，还需进行以下设置。

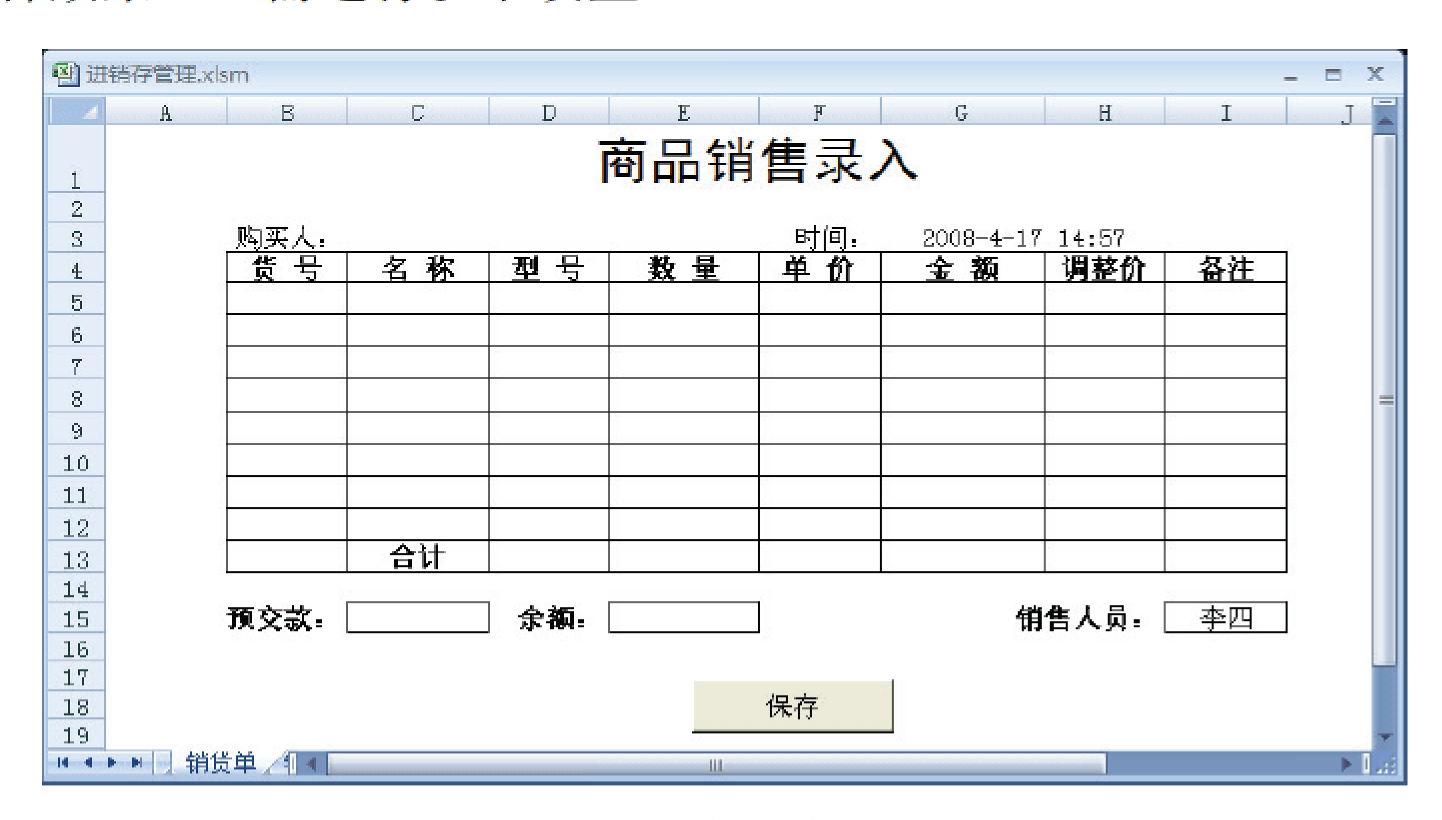


图 31-24　商品销售录入

选中 I15 单元格（销售人员），在【数据】选项卡的【数据工具】组中，单击【数据有效性】按钮，打开【数据有效性】对话框，在【设置】选项卡中设置【来源】选项为“=xsry”（在“销售人员”工作表中定义的名称），如图 31-25 所示。

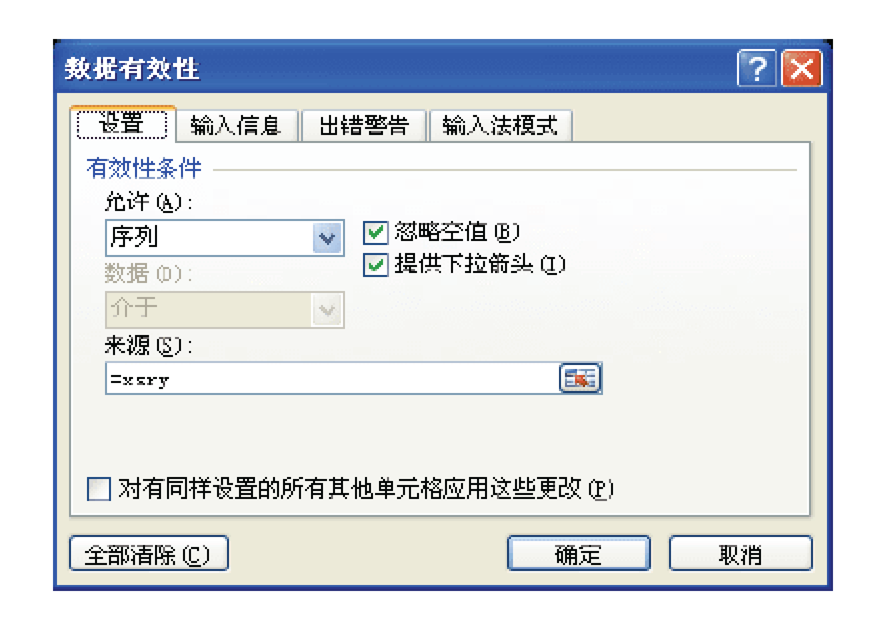


图 31-25　【设置】选项卡

2. 设计公式

在“销货单”中，根据用户输入的货号，系统将自动显示对应的商品信息，但需要为各相关单元格设置公式，具体公式如下：

```
G3=NOW()
```

```
C5=IF($B5="","",VLOOKUP($B5,商品信息!$A:$H,2,FALSE))
D5=IF($B5="","",VLOOKUP($B5,商品信息!$A:$H,3,FALSE))
F5=IF(B5="","",IF($E5<10,VLOOKUP($B5,商品信息!$A:$H,7,FALSE),VLOOKUP($B5,
商品信息!$A:$H,6,FALSE)))
G5=IF(B5="","",IF($H5>0,E5*H5,E5*F5))
G13=SUM(G5:G12)
H13=SUM(H5:H12)
E15=IF(C15=0,"",C15-G13-H13)
```

3. 设计代码

在工作表中输入销售数据后，单击【保存】按钮可将销售数据保存到“销货”工作表中。该按钮的代码如下：

```
Sub 销货输入()
    Dim x As Integer, i As Integer, j As Integer      '声明变量
    Call 手动计算                                      '调用手动计算子过程
    Sheets("销货").Select                              '选择“销货”工作表
    x = 2                                              '从第 2 行开始
    Do While Not (IsEmpty(Cells(x, 2).value))          '判断第 2 列的最后一行
        x = x + 1                                      '在最后一行加一行即为空行
    Loop
    With Sheets("销货单")
        For i = 1 To 8                                 '逐行处理销货单中的数据
            If Not IsEmpty(.Cells(4 + i, 2)) Then      '商品编号不为空
                Sheets("销货").Cells(x, 1) = .Cells(3, 7)  '销售时间
                Sheets("销货").Cells(x, 10) = .Cells(3, 3) '购货人
                Sheets("销货").Cells(x, 11) = .Cells(15, 9) '销售人员
                For j = 2 To 9                         '逐列保存销货信息
                    Sheets("销货").Cells(x, j) = .Cells(4 + i, j)
                Next j
                x = x + 1                              '处理下一行
            End If
        Next i
        .Range("b5:b12") = ""                          '清除销货单中的部分单元格
        .Range("e5:e12") = ""
        .Range("h5:h12") = ""
        .Cells(15, 3) = ""
        .Cells(3, 3) = ""
    End With
    Call 自动计算                                      '调用自动计算子过程
    With Sheets("销货").UsedRange
        .Borders.LineStyle = xlContinuous
        .Borders.Weight = xlThin
    End With
    Sheets("销货").Protect Password:="wyg"
End Sub
```

31.5.2　测试销货单功能

为了检验代码的正确性，需要进行测试，具体步骤如下：

（1）在功能区【进销存】选项卡的【销售】组中，单击【录入】按钮，工作区域将显示“商品销售录入”界面，在表中输入如图 31-26 所示的数据。

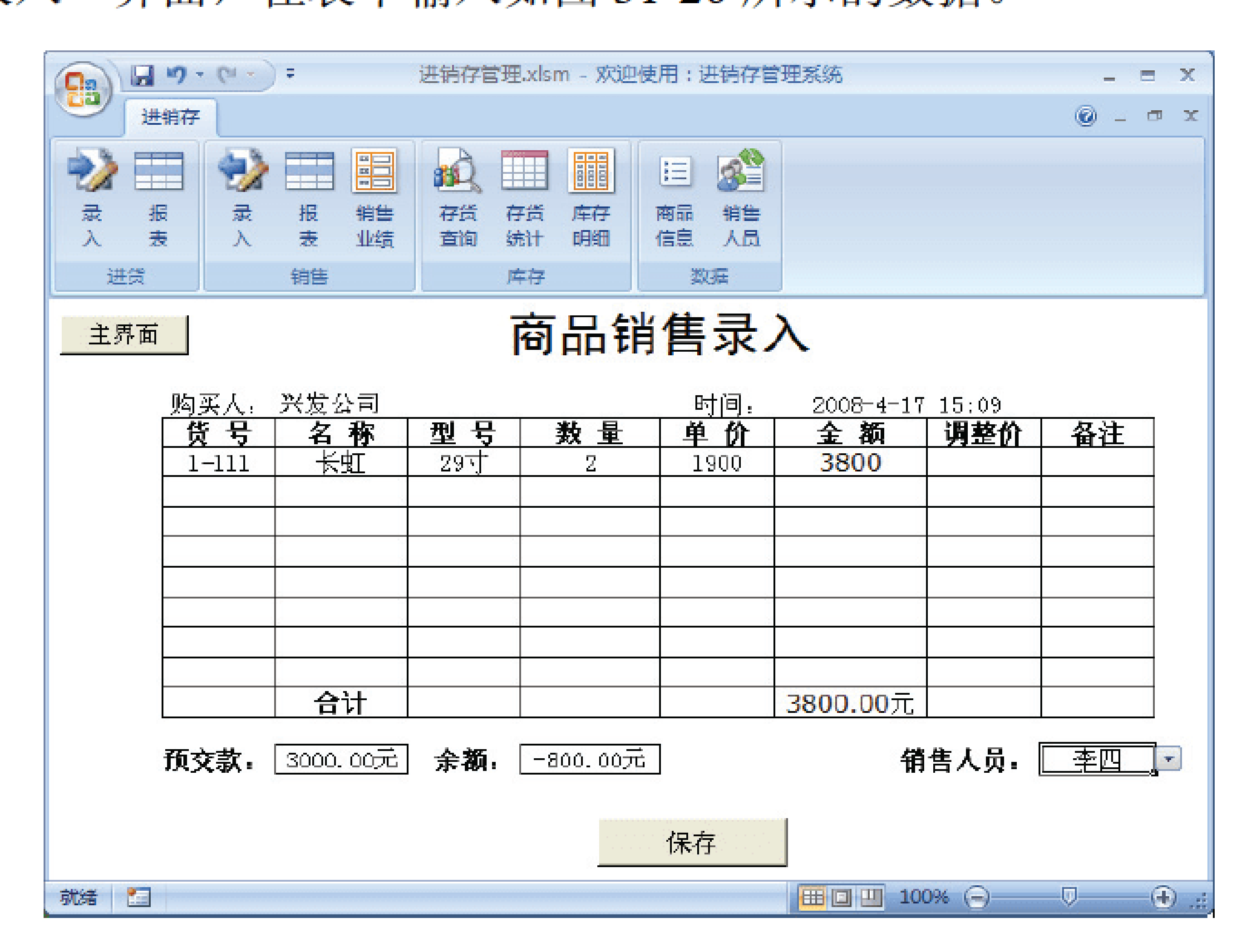


图 31-26　录入数据

（2）输入完成后，单击【保存】按钮，将数据保存到“销货”工作表中，如图 31-27 所示。



图 31-27　销货数据

31.5.3　销售报表

当用户在功能区【进销存】选项卡的【销售】组中，单击【报表】按钮，将显示销售报表。在该报表中由用户输入统计的起止日期，程序将按时间段统计所有进货的情况，并

生成销售报表。

1. 表格设计

首先制作如图 31-28 所示的表头，在“日期”后面的两个单元格输入统计的起止日期，单击【生成报表】按钮，将生成指定时期内的销售报表。

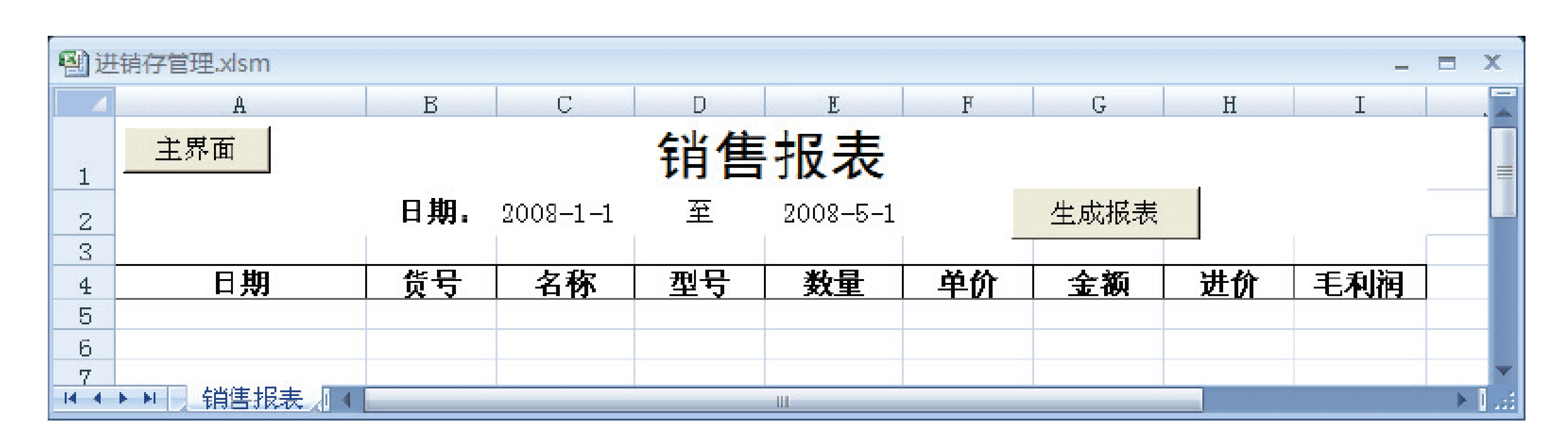


图 31-28 销售报表

2. 编写代码

该报表主要通过 VBA 代码生成，与“进货报表”的代码类似，下面将不再给出程序流程图和程序说明。

（1）为【生成报表】按钮指定宏，并编写宏代码如下：

```
Sub 生成销售报表()
    Dim datStart As Date, datEnd As Date      '声明变量，保存起始和结束日期
    If IsDate(Cells(2, 3)) Then               '起始日期格式正确
        datStart = DateValue(Cells(2, 3))     '保存起始日期到变量
    Else                                      '起始日期格式错误
        MsgBox "日期输入错误"
        Exit Sub                              '退出子过程
    End If
    If IsDate(Cells(2, 5)) Then               '结束日期格式正确
        datEnd = DateValue(Cells(2, 5))       '保存结束日期到变量
    Else                                      '结束日期格式错误
        MsgBox "日期输入错误"
        Exit Sub                              '退出子过程
    End If
    If datStart > datEnd Then                 '起始日期大于结束日期
        MsgBox "起始日期应小于或等于结束日期"  '显示错误信息
    Else
        生成销售报表 a                         '调用子过程生成销售报表
    End If
End Sub
```

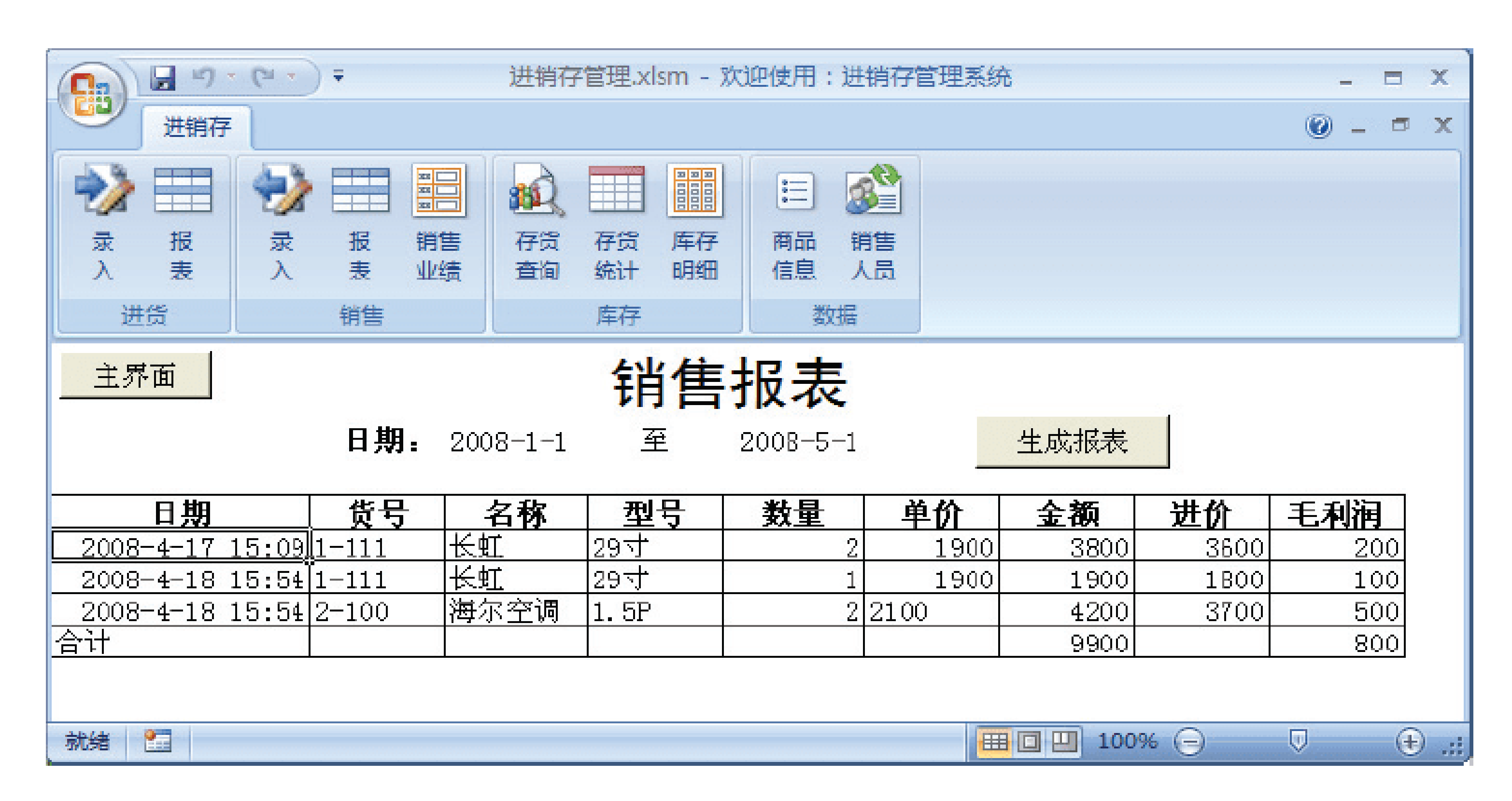
（2）上面的代码中调用了子过程“生成销售报表 a”，其代码如下：

```
Sub 生成销售报表 a()
    Dim x As Integer, j As Integer, rngTemp As Range     '声明变量
    Dim intRow As Integer
    Set rngTemp = Sheets("销货").Range("A2").CurrentRegion
```

```
                                            '获取“销货”工作表的数据区域
rngTemp.Sort Key1:=Sheets("销货").Range("A2"), Order1:=xlAscending, Header:= _
    xlGuess, OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom, _
    SortMethod:=xlPinYin, DataOption1:=xlSortNormal
                                            '将“销货”工作表按货号排序
With Sheets("销售报表")
.Unprotect Password:="wyh"                  '取消对“销售报表”工作表的保护
intRow = .Range("A4").CurrentRegion.Rows.Count  '获取“销售报表”的数据行数
.Range(.Cells(5, 1), .Cells(intRow + 4, 9)).Select
                                            '选择“销售报表”的数据部分
Selection.EntireRow.Delete                  '删除数据部分
.Range("A5").Select                         '选择 A5 单元格
x = 2                                       '销货工作表当前数据行
j = 5                                       '销售报表当前数据行
Do While Not (IsEmpty(Sheets("销货").Cells(x, 2).value))
    If DateValue(Sheets("销货").Cells(x, 1)) >= DateValue(.Cells(2, 3)) And _
       DateValue(Sheets("销货").Cells(x, 1)) <= DateValue(.Cells(2, 5))
       Then
                                            '销货时间大于等于设置的起始时间
                                            '且小于等于结束时间
       .Cells(j, 1).NumberFormatLocal = "yyyy-m-d h:mm;@"
                                            '设置第 1 列为日期格式
       .Cells(j, 1) = Sheets("销货").Cells(x, 1) '日期
       .Cells(j, 2) = Sheets("销货").Cells(x, 2) '货号
       .Cells(j, 3) = Sheets("销货").Cells(x, 3) '名称
       .Cells(j, 4) = Sheets("销货").Cells(x, 4) '型号
       .Cells(j, 5) = Sheets("销货").Cells(x, 5) '数量
       .Cells(j, 6) = Sheets("销货").Cells(x, 6) '单价
       .Cells(j, 7) = Sheets("销货").Cells(x, 7) + Sheets("销货").Cells(x, 8)
                                            '金额
       Sheets("商品查询").Cells(3, 4) = Sheets("销货").Cells(x, 2)
                                            '将货号置入“商品查询”表中查询
       .Cells(j, 8) = Sheets("商品查询").Cells(9, 4)
                                            '从商品查询表中查得进价
      .Cells(j, 9) = .Cells(j, 7) - .Cells(j, 8)    '毛利润
      j = j + 1                             '“销售报表”增加一行
    End If
    x = x + 1                               '处理“销货”工作表的下一行
Loop
.Cells(j, 1) = "合计"                       '在“销售报表”最后一行添加“合计”
.Cells(j, 7).FormulaR1C1 = "=SUM(R[" & 5 - j & "]C:R[-1]C)"
                                            '设置金额合计的计算公式
.Cells(j, 9).FormulaR1C1 = "=SUM(R[" & 5 - j & "]C:R[-1]C)"
                                            '设置毛利润合计的计算公式
.Range(.Cells(5, 1), .Cells(j, 9)).Select   '在“销售报表”中选中数据部分
设置边框                                    '调用子过程设置单元格边框线
.Select                                     '选择“销售报表”工作表
.Range("A5").Select                         '选择 A5 单元格
.Protect Password:="wyh"                    '保护“销售报表”工作表
```

```
    End With
End Sub
```

在“销售报表”日期中输入值，然后单击【生成报表】按钮，得到如图 31-29 所示的销售报表。



日期	货号	名称	型号	数量	单价	金额	进价	毛利润
2008-4-17 15:09	1-111	长虹	29寸	2	1900	3800	3600	200
2008-4-18 15:54	1-111	长虹	29寸	1	1900	1900	1800	100
2008-4-18 15:54	2-100	海尔空调	1.5P	2	2100	4200	3700	500
合计						9900		800

图 31-29　销售报表

31.5.4　销售业绩报表

通过销售业绩可对员工的工作情况进行考核，销售业绩报表就是用来统计指定的时期内某个员工的业绩情况的。

1．表格设计

首先制作如图 31-30 所示的表头，其中“销售员”制作为下拉列表式，“日期”后面的两个单元格用来设置统计的起止日期，单击【生成报表】按钮，将生成某个员工在指定时期内的销售业绩详表。



图 31-30　销售业绩报表

2．编写代码

该报表主要通过 VBA 代码生成，与“进货报表”的代码类似，下面不再给出程序流程图和程序说明。

（1）为“生成报表”按钮指定宏，并编写代码如下：

```
Sub 生成业绩报表()
    Dim datStart As Date, datEnd As Date    '声明变量
    If IsDate(Cells(3, 2)) Then             '检查起始日期
        datStart = DateValue(Cells(3, 2))
    Else
        MsgBox "日期输入错误"
        Exit Sub
    End If
    If IsDate(Cells(3, 5)) Then             '检查结束日期
        datEnd = DateValue(Cells(3, 5))
    Else
        MsgBox "日期输入错误"
        Exit Sub
    End If
    If datStart > datEnd Then               '起始日期大于结束日期
        MsgBox "起始日期应小于或等于结束日期"
        Exit Sub
    End If
    生成业绩报表a                           '调用子过程生成业绩报表
End Sub
```

（2）上段代码调用了“生成业绩报表 a”子过程，其代码如下：

```
Sub 生成业绩报表a()                                 '清除业绩表中的数据
    With Sheets("销售业绩报表")
    .Unprotect Password:="wyh"                      '取消“销售业绩报表”的保护
    intRow = .Range("A5").CurrentRegion.Rows.Count          '获取“销售业绩报
                                                             表”的数据行数
    .Range(.Cells(6, 1), .Cells(intRow + 5, 10)).Select     '选择数据区域部分
    Selection.EntireRow.Delete                      '删除数据区域部分
    .Range("A6").Select                             '选择A6单元格
    x = 2                                           '销货工作表当前数据行
    j = 6                                           '销售业绩报表当前数据行
    Do While Not (IsEmpty(Sheets("销货").Cells(x, 2).value))
        If DateValue(Sheets("销货").Cells(x, 1)) >= DateValue(.Cells(3, 2)) And _
            DateValue(Sheets("销货").Cells(x, 1)) <= DateValue(.Cells(3, 5)) And _
            Sheets("销货").Cells(x, 11) = .Cells(2, 2) Then
                                                   '销货时间大于等于设置的起始时间
                                                   '且小于等于设置的结束时间
                                                   '且销售员等于设置的销售人员
                .Cells(j, 1).NumberFormatLocal = "yyyy-m-d h:mm;@"
                                                   '设置第1列为日期格式
                .Cells(j, 1) = Sheets("销货").Cells(x, 1)  '日期
                .Cells(j, 2) = Sheets("销货").Cells(x, 2)  '货号
                .Cells(j, 3) = Sheets("销货").Cells(x, 3)  '名称
                .Cells(j, 4) = Sheets("销货").Cells(x, 4)  '型号
                .Cells(j, 6) = Sheets("销货").Cells(x, 5)  '数量
                .Cells(j, 7) = Sheets("销货").Cells(x, 6)  '单价
                .Cells(j, 8) = Sheets("销货").Cells(x, 7) + Sheets("销货")
```

```
                    .Cells(x, 8)
                                                              '金额
                    Sheets("商品查询").Cells(3, 4) = .Cells(j, 2)
                                                              '将货号置入“商品查询”表
                                                              '从“商品查询”表中得到商品的信息
                    .Cells(j, 5) = Sheets("商品查询").Cells(8, 4)
                                                              '从“商品查询”表中查得产地
                    .Cells(j, 9) = Sheets("商品查询").Cells(9, 4)
                                                              '从“商品查询”表中查得进价
                    .Cells(j, 10) = .Cells(j, 8) - .Cells(j, 9) * .Cells(j, 6)
                                                              '毛利润=销售金额–进价*数量
                    j = j + 1                                 '销售业绩表增加一行
                End If
                x = x + 1                                     '处理销货表的下一行
            Loop
            If j > 6 Then                                     '若业绩报表有数据
                .Cells(j, 1) = "合计"                          '在最后一行添加合计
                .Cells(j, 8).FormulaR1C1 = "=SUM(R[" & 6 - j & "]C:R[-1]C)"
                                                              '设置数量合计的计算公式
                .Cells(j, 9).FormulaR1C1 = "=SUM(R[" & 6 - j & "]C:R[-1]C)"
                                                              '设置金额合计的计算公式
                .Cells(j, 10).FormulaR1C1 = "=SUM(R[" & 6 - j & "]C:R[-1]C)"
                                                              '毛利润
                                                              '设置毛利润合计的计算公式
                .Range(.Cells(6, 1), .Cells(j, 10)).Select    '选择表格中的数据部分
                设置边框                                        '为数据部分设置单元格边框线
            End If
            .Select                                           '选择销售业绩报表
            .Range("A6").Select                               '选择 A6 单元格
            .Protect Password:="wyh"                          '保护工作表
            End With
End Sub
```

选择销售员“李四”，再输入起止日期，单击【生成报表】按钮，将得到如图 31-31 所示的销售业绩报表。

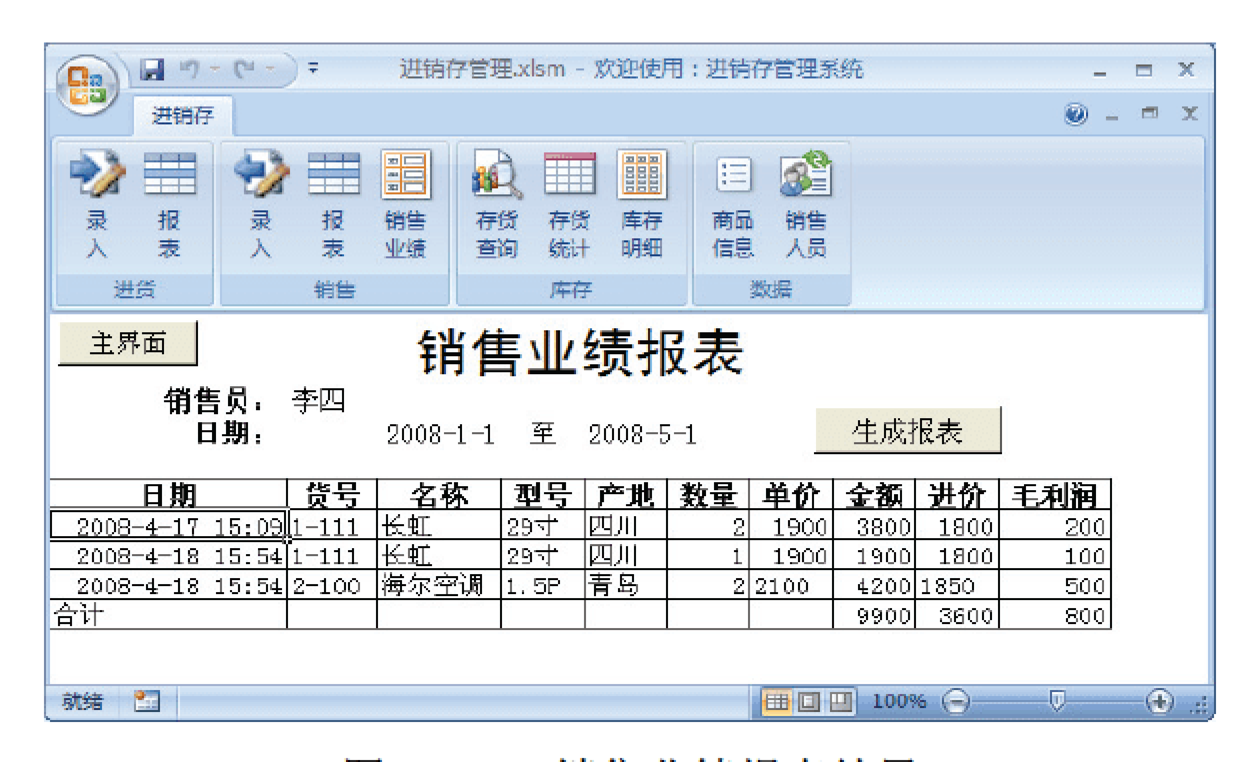


图 31-31 销售业绩报表结果

31.6　库 存 模 块

库存模块用来查询生成当前库存数据，包括按货号查询单一商品信息的库存查询、查看所有库存商品的库存统计和按货号查询单一商品进、销、存数据的库存明细等功能。

31.6.1　商品查询

通过商品查询功能可查看某一货号的商品的供货、销货及库存情况，该表不需要编写 VBA 代码，通过 Excel 的公式即可完成相应的功能。具体步骤如下：

（1）首先制作如图 31-32 所示的工作表，用户可在货号位置输入内容。

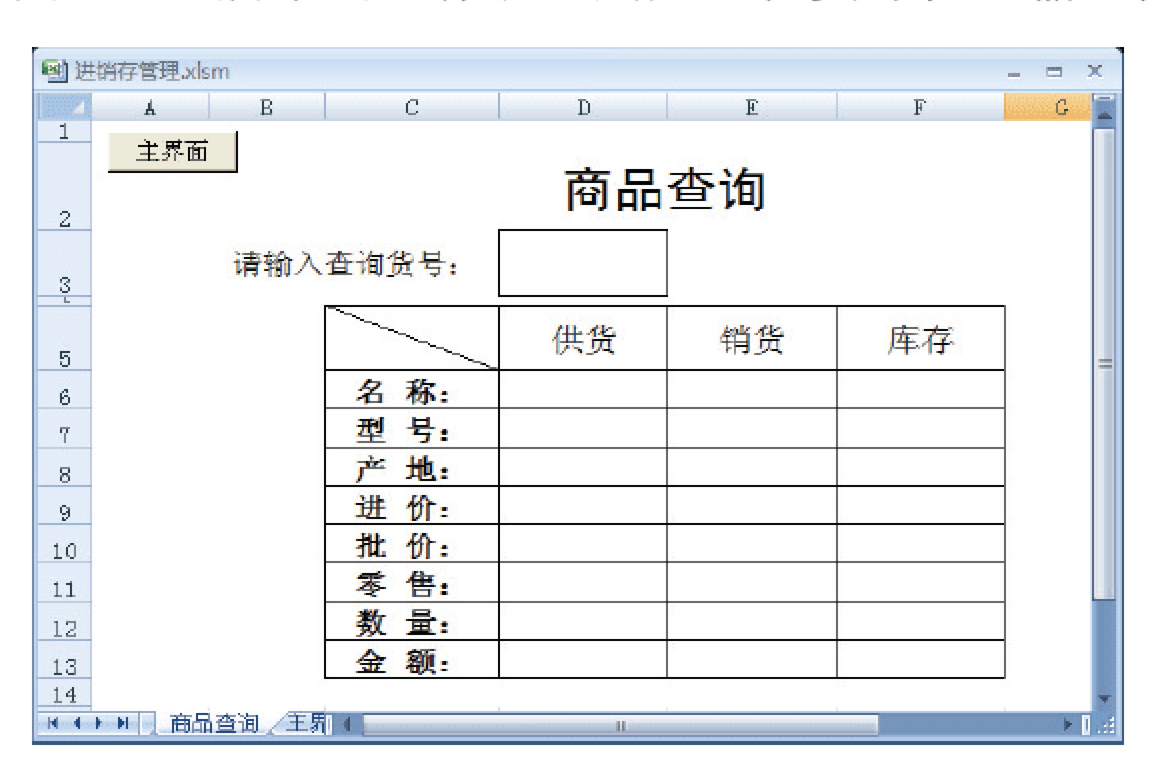


图 31-32　商品查询

（2）设置各单元格的公式如下：

```
D6 =VLOOKUP($D$3,供货!$B:$J,2,FALSE)
D7=VLOOKUP($D$3,供货!$B:$J,3,FALSE)
D8=VLOOKUP($D$3,供货!$B:$J,4,FALSE)
D9=VLOOKUP($D$3,供货!$B:$J,5,FALSE)
D10=VLOOKUP($D$3,供货!$B:$J,6,FALSE)
D11=VLOOKUP($D$3,供货!$B:$J,7,FALSE)
D12=SUMIF(供货!B:B,D3,供货!I:I)
E12=SUMIF(销货!B:B,D3,销货!E:E)
F12=D12-E12
D13=D9*D12
E13=E9*E12
F13=D13-E13
E6、F6=D6
E7、F7=D7
E8、F8=D8
E9、F9=D9
E10、F10=D10
E11、F11=D11
```

在货号中输入“1-111”，得到如图 31-33 所示内容。

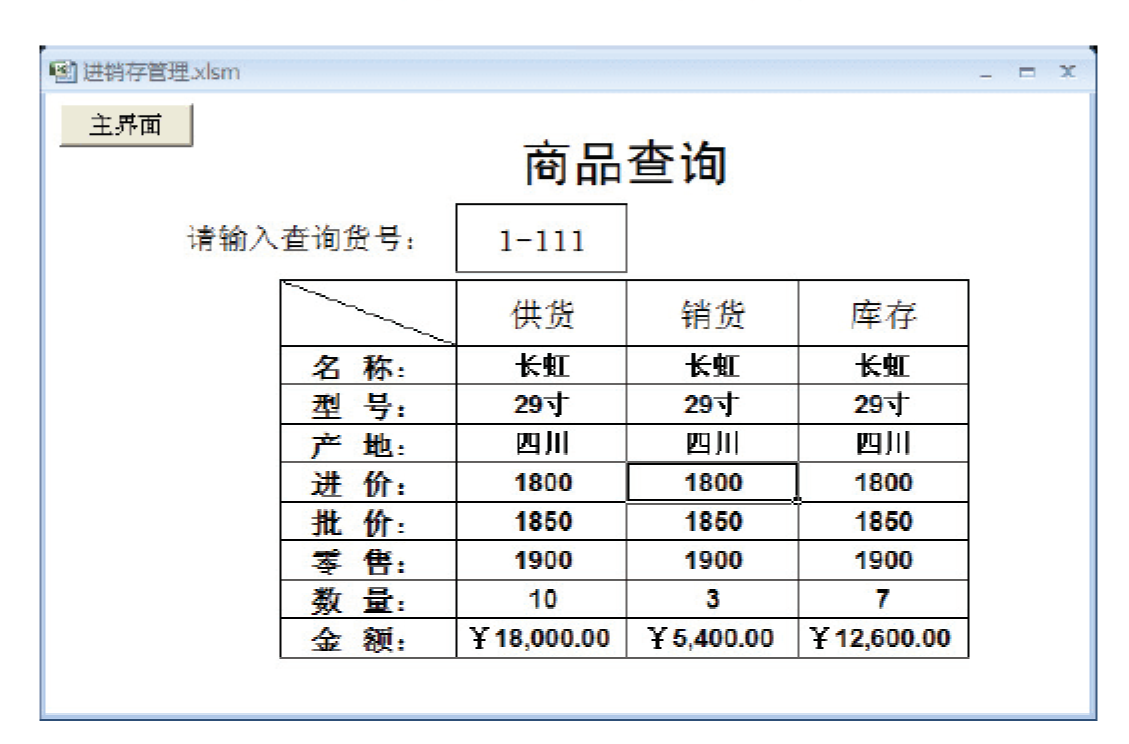


图 31-33　商品查询结果

31.6.2　存货统计

31.6.1 节制作的商品查询模块可查询单件商品的信息，本节以此为基础，编写商品存货统计功能。首先从“商品信息”中取出一件商品的货号，将该货号置入“商品查询”表的货号位置，然后从“商品查询”工作表中获取对应的信息，并写入“存货统计”工作表中。对整个“商品信息”工作表进行循环，最后得到“存货统计”表。在“存货统计”工作表的 Activate 事件中编写代码如下：

```
Private Sub Worksheet_Activate()
   Dim i As Integer, x As Integer                         '声明变量
   Sheets("商品信息").Select                              '选择“商品信息”工作表
   x = 2                                                  '从第 2 行开始
   Do While Not (IsEmpty(Cells(x, 1)))                    '判断第 2 列的最后一行
      x = x + 1                                           '在最后一行加一行即为空行
   Loop
   x = x - 1                                              '返回上一行
   With Sheets("商品查询")
      For i = x To 2 Step -1
         .Cells(3, 4) = Sheets("商品信息").Cells(i, 1)      '货号
         Sheets("存货统计").Cells(i, 1) = .Cells(3, 4)      '货号
         Sheets("存货统计").Cells(i, 2) = .Cells(6, 4)      '名称
         Sheets("存货统计").Cells(i, 3) = .Cells(12, 4)     '总进量
         Sheets("存货统计").Cells(i, 4) = .Cells(12, 5)     '总出量
         Sheets("存货统计").Cells(i, 5) = .Cells(12, 6)     '库存量
         Sheets("存货统计").Cells(i, 6) = .Cells(9, 4)      '进价
         Sheets("存货统计").Cells(i, 7) = .Cells(13, 6)     '资金
      Next
   End With
   Sheets("存货统计").Select                              '选择“存货统计”工作表
End Sub
```

在【进销存】选项卡的【库存】组中，单击【存货统计】按钮，程序在经过一段时间的运算后，产生“存货统计”工作表中的数据，如图 31-34 所示。



货号	名称	总进量	总出量	库存量	单价	金额	备注
1-111	长虹	10	3	7	1800	12600	
1-113	长虹空调	5		5	1500	7500	
1-112	长虹	5		5	2400	12000	
1-121	长虹冰箱	10		10	1500	15000	
2-111	海尔冰箱	15		15	1400	21000	
2-100	海尔空调	10	2	8	1850	14800	

图 31-34　存货统计

31.6.3　库存明细

在 31.6.1 节中设计制作的商品查询模块只能看到经过汇总的数据。如果想了解某个商品的详细进、销、存数据，可使用本模块。

1．表格设计

本模块将能显示每件商品的进、销、存明细数据，并模拟商品账页形式进行显示，工作表如图 31-35 所示。输入货号后，下方自动显示商品名称等信息，设置好起止日期后，单击【生成账页】按钮，将生成指定商品的详细数据。

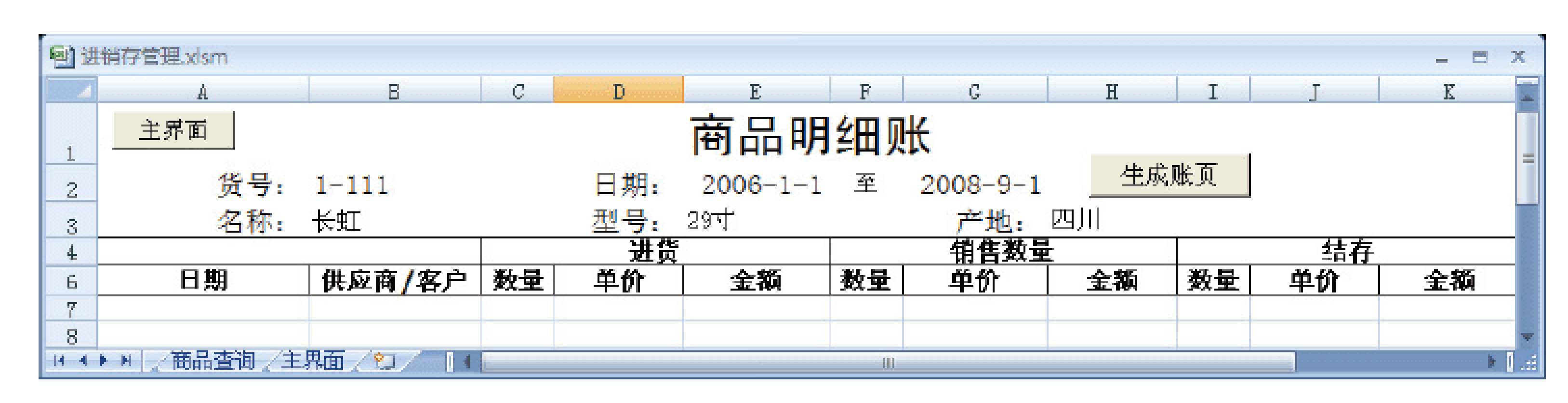


图 31-35　商品明细账

2．设计公式

在表格中，输入货号后，将通过 VLOOKUP 公式，显示出对应货号的商品信息，设置公式如下：

```
B3=VLOOKUP($B$2,供货!$B:$J,2,FALSE)
E3=VLOOKUP($B$2,供货!$B:$J,3,FALSE)
H3=VLOOKUP($B$2,供货!$B:$J,4,FALSE)
```

3．设计代码

该报表主要通过 VBA 代码生成，首先读入进货数据和销货数据，然后通过设置公式计算出存货数据，具体代码如下。

（1）为【生成账页】按钮指定宏，并编写代码如下：

```
Dim j As Integer                              '声明模块级变量，保存报表当前行
Sub 生成商品明细账()
```

```
    Dim intRow As Integer                        '声明变量，保存表格行数
    Dim datStart As Date, datEnd As Date         '声明变量，保存起始、结束日期
    If IsDate(Cells(2, 5)) Then                  '检查起始日期
        datStart = DateValue(Cells(2, 5))
    Else
        MsgBox "日期输入错误"
        Exit Sub
    End If
    If IsDate(Cells(2, 7)) Then                  '检查结束日期
        datEnd = DateValue(Cells(2, 7))
    Else
        MsgBox "日期输入错误"
        Exit Sub
    End If
    If datStart > datEnd Then                    '检查起始日期是否大于结束日期
        MsgBox "起始日期应小于或等于结束日期"
        Exit Sub
    End If
    Sheets("商品明细账").Unprotect Password:="wyh"
                                                 '取消“商品明细账”工作表的保护
    With Sheets("商品明细账")
        intRow = .Range("A5").CurrentRegion.Rows.Count
                                                 '获取“商品明细账”中的数据行数
        .Range(.Cells(7, 1), .Cells(intRow + 5, 10)).Select
                                                 '选择报表的数据部分
        Selection.EntireRow.Delete               '清除商品明细账中的数据
        .Range("A7").Select
    End With
    读入进货数据                                 '调用子过程获取进货数据
    读入销售数据                                 '调用子过程获取销售数据
    计算存货                                     '调用子过程计算库存数据
End Sub
```

程序中首先设置模块级变量 j，用来记录生成的“商品明细账”数据行数。因为在该子过程中将调用几个子过程对“商品明细账”中的数据进行处理，所以通过模块级变量，可共享正在处理的行数。

（2）“读入进货数据”子过程的功能是从“供货”工作表中逐行读入数据进行判断，并将满足要求的数据写入“商品明细账”工作表中。其详细代码如下：

```
Sub 读入进货数据()
    Dim x As Integer                       '声明变量
    With Sheets("商品明细账")
    x = 2                                  '供货表的当前行
    j = 7                                  '商品明细账的当前行(模块级变量)
    Do While Not (IsEmpty(Sheets("供货").Cells(x, 2).value))
        If DateValue(Sheets("供货").Cells(x, 1)) >= DateValue(.Cells(2, 5)) And _
            DateValue(Sheets("供货").Cells(x, 1)) <= DateValue(.Cells(2, 7)) And _
            Sheets("供货").Cells(x, 2) = .Cells(2, 2) Then
                                           '供货日期大于等于设置的起始日期
                                           '且小于等于设置的结束日期
                                           '且供货表中的货号等于设置的货号
```

```
            .Cells(j, 1).NumberFormatLocal = "yyyy-m-d h:mm;@"
                                          '设置第 1 列为日期格式
            .Cells(j, 1) = Sheets("供货").Cells(x, 1)      '日期
            .Cells(j, 2) = Sheets("供货").Cells(x, 11)     '供应商
            .Cells(j, 3) = Sheets("供货").Cells(x, 9)      '数量
            .Cells(j, 4).NumberFormatLocal = "#,##0.00_ "
                                          '设置进价的显示格式
            .Cells(j, 4) = Sheets("供货").Cells(x, 6)      '进价
            .Cells(j, 5).NumberFormatLocal = "#,##0.00_ "
                                          '设置金额的显示格式
            .Cells(j, 5) = .Cells(j, 3) * .Cells(j, 4)    '金额
            j = j + 1                     '商品明细账增加一行
        End If
        x = x + 1                         '处理供货表中的下一行数据
    Loop
    End With
End Sub
```

该子过程比较简单，不再画流程图。程序中主要对“供货”工作表中的数据进行判断，如果数据在指定的日期内，且为指定的货号，则将数据复制到“商品明细账”工作表的进货单元格中。

（3）“读入销售数据”子过程的功能是从“销货”工作表中逐行读入数据进行判断，并将满足要求的数据写入“商品明细账”工作表中。其详细代码如下：

```
Sub 读入销售数据()
    Dim x As Integer                      '声明变量
    With Sheets("商品明细账")
    x = 2                                 '销货工作表当前数据行
    Do While Not (IsEmpty(Sheets("销货").Cells(x, 2).value))
        If DateValue(Sheets("销货").Cells(x, 1)) >= DateValue(.Cells(2, 5)) And _
            DateValue(Sheets("销货").Cells(x, 1)) <= DateValue(.Cells(2, 7)) And _
            Sheets("销货").Cells(x, 2) = .Cells(2, 2) Then
                                          '销货日期大于等于设置的起始日期
                                          '且小于等于设置的结束日期
                                          '且销货货号等于设置的货号
            .Cells(j, 1).NumberFormatLocal = "yyyy-m-d h:mm;@"
                                          '设置第 1 列为日期格式
            .Cells(j, 1) = Sheets("销货").Cells(x, 1) '日期
            .Cells(j, 2) = Sheets("销货").Cells(x, 10)    '购货人
            .Cells(j, 6) = Sheets("销货").Cells(x, 5) '数量
            .Cells(j, 7).NumberFormatLocal = "#,##0.00_ "
                                          '设置单价的显示格式
            .Cells(j, 7) = Sheets("销货").Cells(x, 6) '单价
            .Cells(j, 8).NumberFormatLocal = "#,##0.00_ "
                                          '设置金额的显示格式
            .Cells(j, 8) = Sheets("销货").Cells(x, 7) + Sheets("销货").Cells(x, 8)
                                          '金额
           j = j + 1                      '商品明细账增加一行
        End If
        x = x + 1                         '处理销货的下一行
```

```
    Loop
    End With
End Sub
```

（4）“计算存货”子过程的流程图如图 31-36 所示。

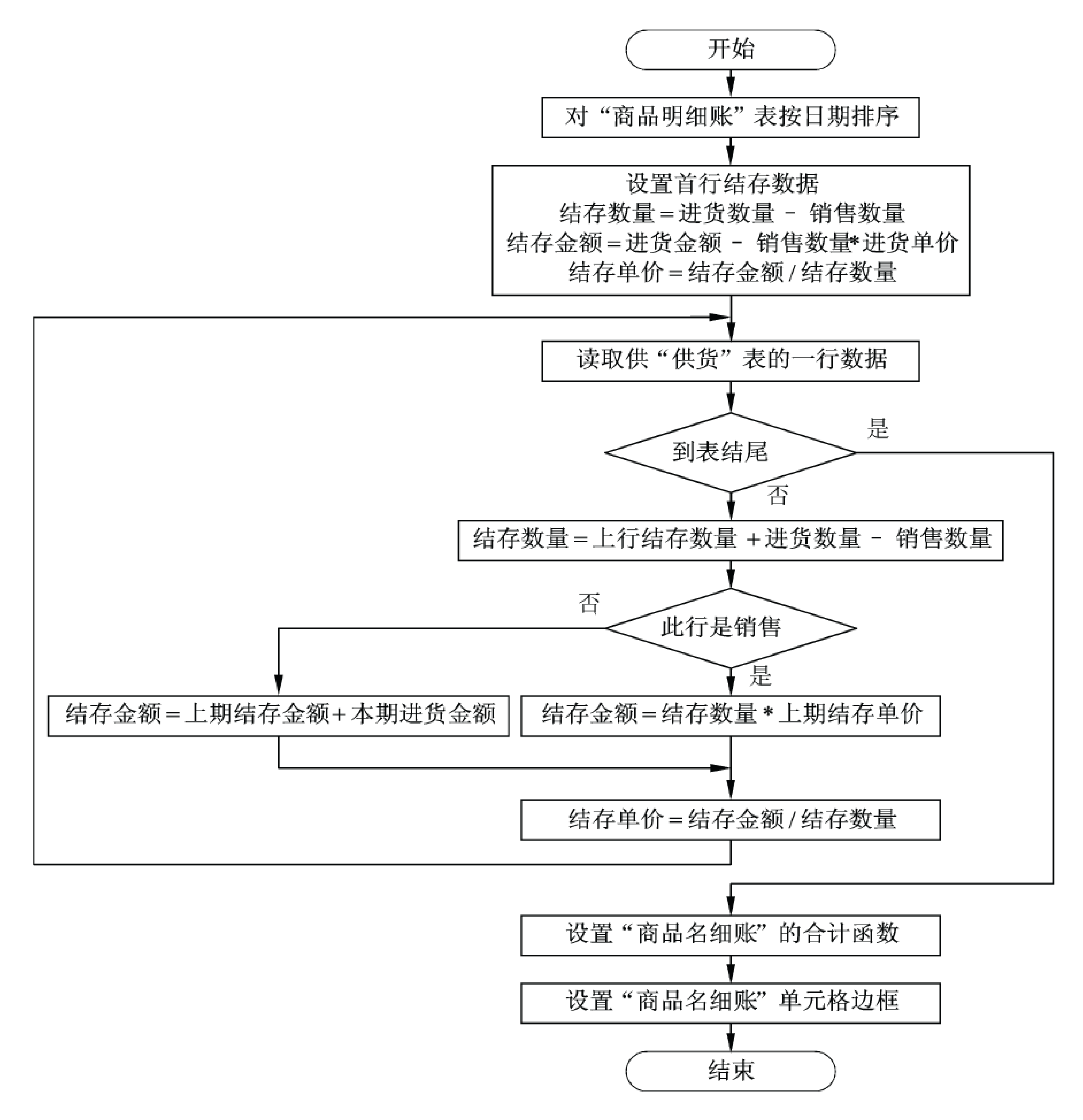


图 31-36　计算存货流程图

通过“读入进货数据”和“读入销售数据”两个子过程处理后，“商品明细账”中已经有了需要的全部数据，但数据的排列方式为全部显示完进货数据后，再显示销售数据，这样显示不符合账页的要求，这时将所有数据按日期排序之后即可。

接着通过“结存数量=上期结存数量+进货数量–销售数量”公式，计算出结存数量。

根据流程图编写“计算存货”子过程的代码如下：

```
Sub 计算存货()
    Dim x As Integer, rngTemp As Range
    With Sheets("商品明细账")
    Set rngTemp = .Range("A7").CurrentRegion     '获取商品明细账的数据区域
```

```
    rngTemp.Sort Key1:=.Range("A7"), Order1:=xlAscending, Header:= _
        xlGuess, OrderCustom:=1, MatchCase:=False, Orientation:=
        xlTopToBottom, _
        SortMethod:=xlPinYin, DataOption1:=xlSortNormal '对该区域按日期进行排序
    j = j - 1                                   '将模块级变量 j 减 1，获得数据行数
    .Cells(7, 9) = .Cells(7, 3) - .Cells(7, 6)  '结存数量 = 进货数量 – 销售数量
    .Cells(7, 11).NumberFormatLocal = "#,##0.00_ " '设置金额单元格显示格式
    .Cells(7, 11) = (.Cells(7, 3) - .Cells(7, 6)) * .Cells(7, 4)
                                                '结存金额=结存数量*进货单价
    .Cells(7, 10).NumberFormatLocal = "#,##0.00_ " '设置单价单元格显示格式
    .Cells(7, 10) = .Cells(7, 11) / .Cells(7, 9)
                                                '单价=结存金额 / 结存数量
    For x = 8 To j
        .Cells(x, 9) = .Cells(x - 1, 9) + .Cells(x, 3) - .Cells(x, 6)
                                                '结存数量
        .Cells(x, 11).NumberFormatLocal = "#,##0.00_ " '设置金额单元格显示格式
        If IsEmpty(.Cells(x, 3)) Then           '若进货数量为空，则为销售
            .Cells(x, 11) = .Cells(x, 9) * .Cells(x - 1, 10)
                                                '结存金额=结存数量*上行的单价
        Else '若进货数量不为空，则为进货
            .Cells(x, 11) = .Cells(x - 1, 11) + .Cells(x, 5)
                                                '结存金额=上行结存金额+进货金额
        End If
        .Cells(x, 10).NumberFormatLocal = "#,##0.00_ " '设置结存单价显示格式
        .Cells(x, 10) = .Cells(x, 11) / .Cells(x, 9)   '结存单价=结存金额 /
                                                        结存单价
    Next
    .Cells(x, 1) = "合计"                              '最后一行设为“合计”
    .Cells(x, 3) = "=SUM(R[" & 7 - x & "]C:R[-1]C)" '设置进货数量合计的计算公式
    .Cells(x, 5).NumberFormatLocal = "#,##0.00_ "  '设置进货金额合计的显示格式
    .Cells(x, 5).FormulaR1C1 = "=SUM(R[" & 7 - x & "]C:R[-1]C)"
                                                   '设置进货金额合计的计算公式
    .Cells(x, 6) = "=SUM(R[" & 7 - x & "]C:R[-1]C)" '设置销售数量合计的计算公式
    .Cells(x, 8).NumberFormatLocal = "#,##0.00_ "  '设置销售金额合计的显示格式
    .Cells(x, 8).FormulaR1C1 = "=SUM(R[" & 7 - x & "]C:R[-1]C)"
                                                   '设置销售金额合计的计算公式
    .Cells(x, 9) = "=SUM(R[" & 7 - x & "]C:R[-1]C)" '设置结存数量合计的计算公式
    .Cells(x, 11).NumberFormatLocal = "#,##0.00_ " '设置结存金额合计的显示格式
    .Cells(x, 11).FormulaR1C1 = "=SUM(R[" & 7 - x & "]C:R[-1]C)"
                                               '设置结存金额合计的计算公式
    .Cells(x, 10).NumberFormatLocal = "#,##0.00_" '设置结存单价的显示格式
    .Cells(x, 10) = .Cells(x, 11) / .Cells(x, 9)  '结存单价=结存金额/结存数量
    .Range(.Cells(7, 1), .Cells(x, 11)).Select  '选择“商品明细账”的数据部分
    设置边框                                    '设置单元格的边框线
    .Select                                     '选择“商品明细账”工作表
    .Range("A7").Select                         '选择 A7 单元格
    End With
    Sheets("商品明细账").Protect Password:="wyh" '保护“商品明细账”工作表
End Sub
```

输入货号，设置好起始日期后，单击【生成账页】按钮，生成的商品明细账如图 31-37

所示。

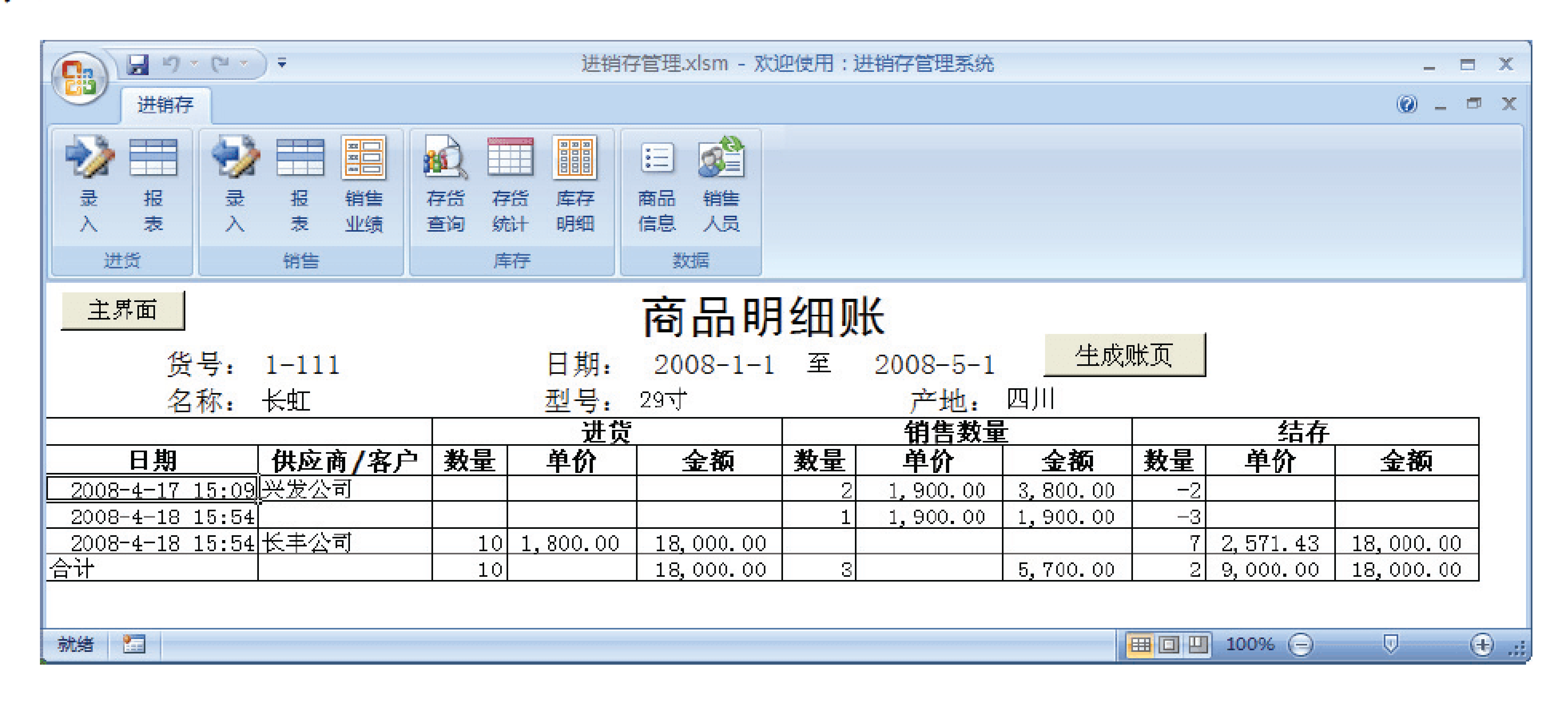


图 31-37　商品明细账

在【进销存】选项卡的【数据】组中的两个按钮用来设置辅助数据，这两个工作表的数据不需要编写 VBA 代码，由用户直接输入数据即可。

附录 A　VBA 程序调试技巧

对于一个应用程序，开发人员将花费很多时间在调试工作上。有时按设计要求编写的一段代码，通常都包含着很多错误，这时就需要通过调试来找出代码中的错误，并将这些错误修复，使代码最终能够实现预期的功能。

A.1　VBA 程序的模式

VBA 程序有 3 种模式：设计时、运行时和中断模式。为了测试和调试一个程序，需要了解在当前时间程序处于这 3 种模式中的哪一种。

在设计时使用 VBA 可编写应用程序的代码。而在运行时是运行应用程序。在中断模式下，程序的执行被暂停，用户能够检测和修改相关变量的数据。在中断模式时，VBE 标题栏中将显示“[中断]”的提示文字。

下面列出这 3 种模式的特征以及在这 3 种模式间转换的技巧。

- 设计时：创建应用程序的大部分工作是在设计时完成的。可设计窗体、编写代码。可设置断点和创建监视表达式，不能运行代码或使用调试工具。要切换到运行时，可单击标准工具栏中的【运行子过程/用户窗体】按钮（或按 F5 键）。要切换到中断模式，可单击菜单【调试】|【逐语句】命令（或按 F8 键）。
- 运行时：在应用程序控制下，可与应用程序间进行交互。此时可查看程序代码，但不能修改代码。要切换回设计时，可单击工具栏中的【重新设置】按钮。要切换到中断模式，可单击【中断】按钮。
- 中断模式：在运行应用程序时暂停执行。可在同一点查看和编辑代码，检测或修改数据，重新启动应用程序，结束运行或继续运行。要切换到运行时，可单击工具栏中的【继续】按钮（在中断模式中，【运行】按钮变成了【继续】按钮）。要切换到设计时，可单击工具栏中的【重新设置】按钮。

可在设计时设置断点和监视表达式，但其他调试工具仅在中断模式下工作。

在 VBA 代码执行过程中，按 ESC 键可终止执行，将程序挂起，并打开如图 A-1 所示的对话框。



图 A-1　程序挂起

在图 A-1 所示对话框中有 4 个按钮，其作用分别为以下 4 个方面。

- 继续：单击该按钮可以恢复代码执行。如果遇到错误，该按钮将变为禁止使用状态。
- 结束：如果这次不想排除故障，则单击该按钮，

VBA 将终止代码执行。

- ❑ 调试：单击该按钮进入中断模式，代码窗口将出现，VBE 将加亮过程执行时停止处的代码行。可以检查、调试、中断或者逐句执行代码。
- ❑ 帮助：单击该按钮查看在线帮助，解释导致该错误信息的原因。

A.2 设 置 断 点

在设计和中断模式下都可以设置断点，当程序执行到断点语句时，则中断执行，这时处于中断模式。

如果事前预料到某部分代码将出现问题，可在该代码前面暂停执行代码，然后通过逐语句执行，发现出问题的具体代码。

例如，在第 31 章的例子中，在【工程】资源管理器窗口中双击“存货统计”工作表，打开该工作表的事件代码。在代码窗口中找到要设置断点的位置，单击菜单【调试】|【快速断点】命令（或按 F9 键）设置断点。VBE 将在页边的空白处显示一个红色的圈，并将设置断点的代码用红底白字显示出来，如图 A-2 所示。

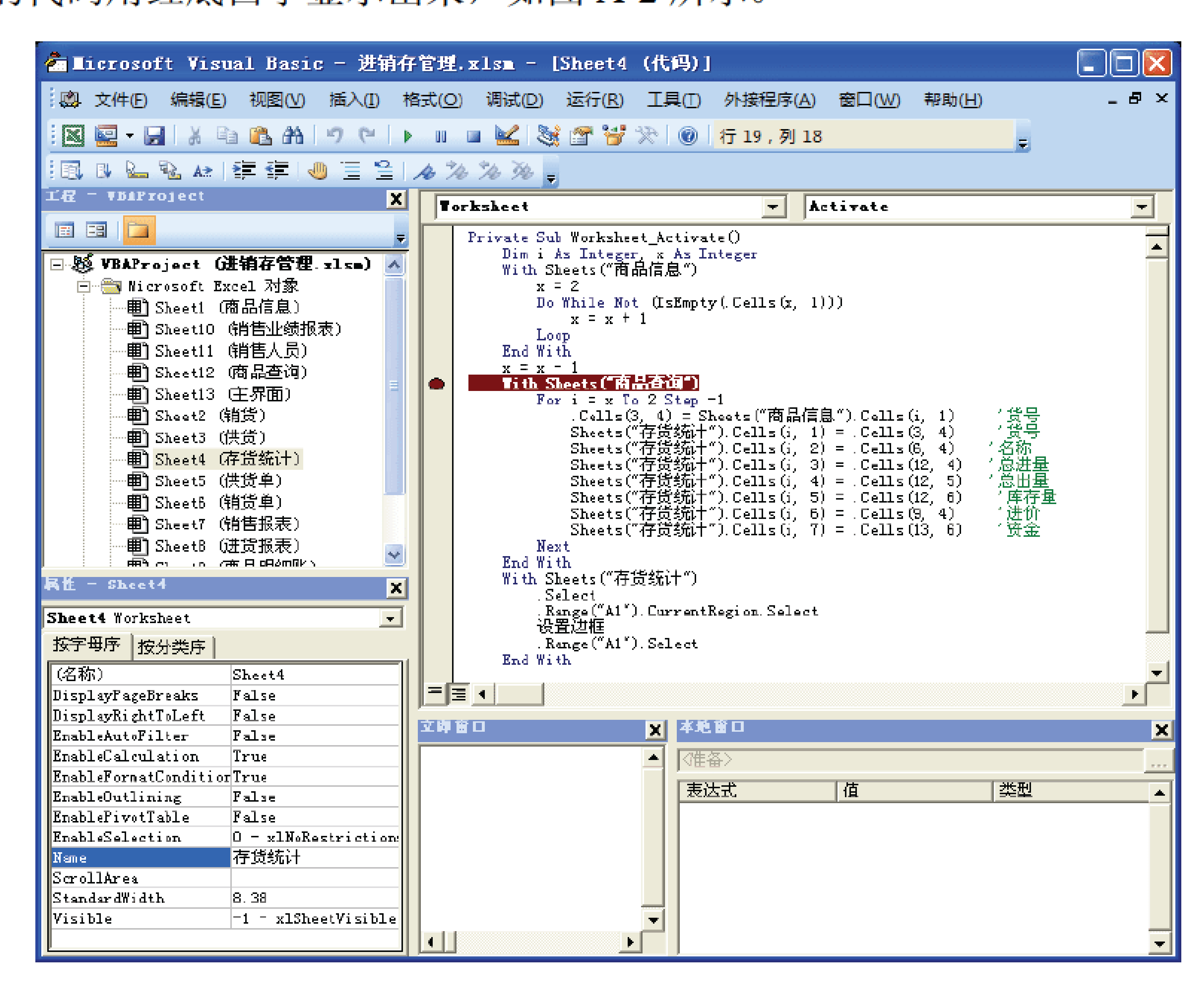


图 A-2 设置断点

🔔**技巧**：用鼠标单击代码窗口左侧的页边，可快速地设置断点。

设置好断点后，返回 Excel。在【进销存】选项卡的【库存】组中，单击【存货统计】按钮，将执行设置断点处的代码，当执行到断点处将中断程序执行。在中断的代码窗口中，将要被执行的语句左侧页边上显示一个黄色箭头，语句显示为黄底黑字。

这时，可多次选择菜单【调试】|【逐语句】命令（或按 F8 键），逐条语句执行断点下方的代码，观察是否有出错地方。

注意：过程运行结束后，设置的断点语句仍然是加亮显示的，VBA 不会自动删除断点。

通过单击菜单【调试】|【清除所有断点】命令（或按 Shift+Ctrl+F9 组合键）将清除程序中所有的断点。

技巧：关闭工作簿后，设置的所有断点都将自动清除。

A.3　代码调试运行方式

当程序进行中断模式后，可通过多种方式执行一条语句、一个过程或执行指定区域的代码，这些命令都在【调试】菜单中提供。通过这些方式执行部分 VBA 代码，以了解程序执行的路径，观察代码执行的顺序是否与设计过程一致。

常用的调试运行方式有以下几种。

- 逐语句：一次执行一个语句（按程序流程逐个语句执行）。
- 逐过程：与“逐语句”相似。只有在当前的语句含有一个对过程的调用时，两者才会有差异。“逐过程”是将过程视为一个基本单位来执行，执行完一个语句后再继续执行下一个语句。不过，下个被显示的语句，就是当前过程中的下一个语句，不会因为当前语句为一过程调用而有所改变。
- 跳出：执行当前执行点所在函数中剩余未执行的行。下个被显示的语句是紧随在该过程调用后的语句。所有在当前与最后的执行点间的代码都会被执行。
- 运行到光标处：当应用程序处于中断模式时，可以在代码中选定想执行到哪一行语句才停止，然后单击菜单【调试】|【运行到光标处】命令，这样，应用程序将会从当前语句执行到选定的语句，从而“跳过”不感兴趣的代码部分，例如，大型循环或认为不会出错的大部分语句等。
- 设置下一条语句：在调试或试验一个应用程序时，可以在当前过程中的任意行选定一个语句，然后选择菜单【调试】|【设置下一条语句】命令来跳过某些代码段（例如，包含已知错误的代码段）。这样，可继续跟踪其他问题。或者想返回到当前语句前面的语句，以便使用不同的变量值或属性值对那部分代码进行测试。
- 显示下一条语句：单击菜单【调试】|【显示下一条语句】命令，可将光标移到下一行会被执行的程序。如果已在执行一个错误处理程序的代码但不能确认将执行的下一语句，这个特征是很方便的。

A.4　监视表达式

程序中的许多错误是由变量获得未预期的值而引用的，如出现程序最终结果不正确的

逻辑错误，可能是变量的值在运算过程中出了问题。如果程序运行时需要变量的值为数值型，但变量为字符，也将中断程序。

如果过程中使用了一个值经常变化的变量，在调试程序时，暂停程序查看该变量是否为需要的值，可排查程序的逻辑错误。VBA 提供了一个特别的监视窗口，通过该窗口的设置可监视变量或表达式在运行时的值。

1. 添加监视

添加监视表达式的步骤如下：

（1）在代码窗口中选择准备监视的变量。

（2）单击菜单【调试】|【添加监视】命令，打开【添加监视】对话框，如图 A-3 所示。

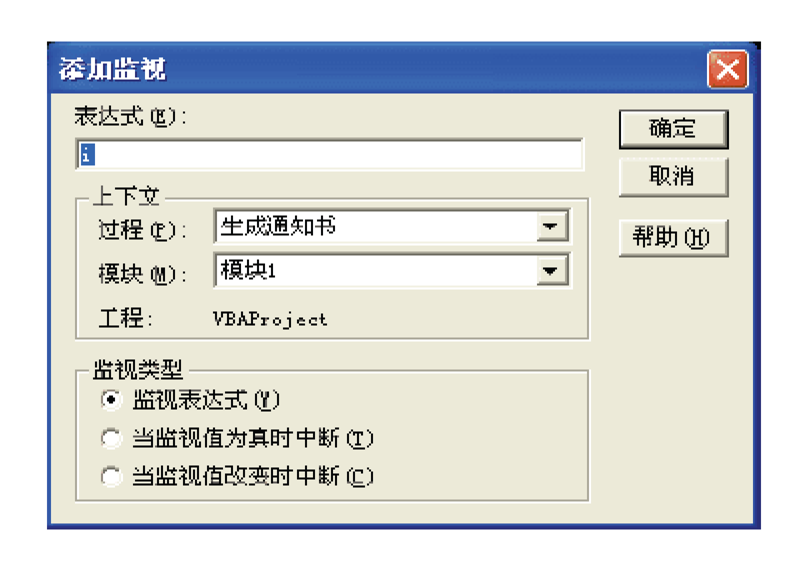


图 A-3 【添加监视】对话框

【添加监视】对话框包含下述 3 部分内容。

- ❑ 表达式：显示需要监视的变量或表达式（也可在该文本框中输入）。
- ❑ 上下文：指定包含该变量的过程名和模块名。
- ❑ 监视类型：设置监视变量的方式。选中【监视表达式】选项按钮，在中断模式时能够在监视窗口查看该变量的值。选中【当监视值为真时中断】选项按钮，当监视变量的值为真（非零）时 VBA 将中断过程。选中【当监视值改变时中断】选项按钮，只要程序执行改变了监视变量或表达式的值，程序就会停止运行。

技巧：选中【当监视值为真时中断】选项按钮时，一般将监视表达式定义为一个逻辑表达式，如“i=5”。

（3）单击【确定】按钮，添加一个监视表达式到【监视窗口】列表框中，如图 A-4 所示。

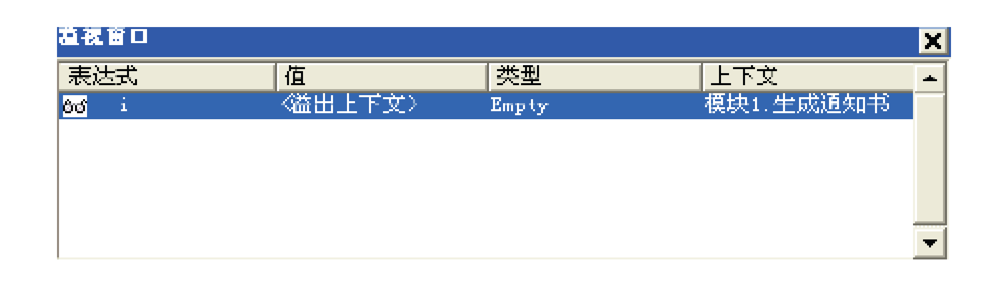


图 A-4 监视窗口

技巧：也可以在过程执行中断之后添加监视表达式。

在图 A-4 所示的【监视窗口】列表框中单击选中一个监视表达式，按 Delete 键可将该监视表达式清除。

2. 使用快速监视

在没有定义监视表达式时，如果要查看一个表达式的值，可以使用快速监视。具体操作方法如下：

（1）在中断模式下，将光标放在变量名或表达式内部。

（2）单击菜单【调试】|【快速监视】命令（或按 Shift+F9 组合键），将打开如图 A-5 所示的【快速监视】对话框。

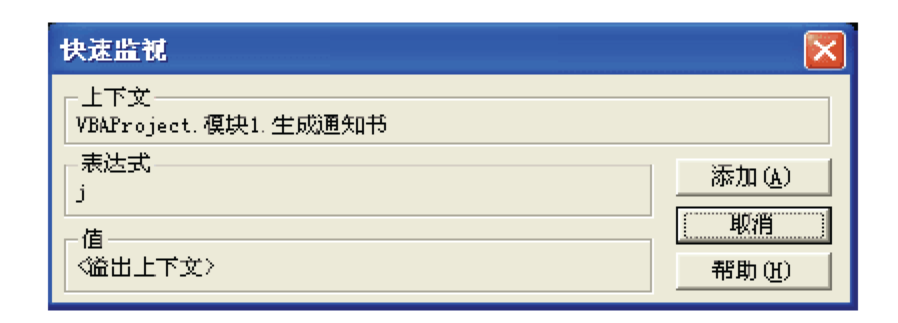


图 A-5 【快速监视】对话框

（3）单击【添加】按钮，可将表达式添加到【监视窗口】列表框中。

A.5　使用本地窗口

【本地窗口】列表框显示当前过程范围内所有变量的值。当执行由一个过程切换到另一过程时，【本地窗口】列表框的内容将转变为仅反应适用于当前过程的变量。

【本地窗口】列表框如图 A-6 所示，共包括下述 3 列。

- ❑ 表达式：显示当前过程里声明的变量名称。第一行显示的是“模块”变量，单击前面的加号，就可以查看是否定义有模块级变量。

注意：全局变量不会在“本地”窗口中显示。

- ❑ 值：在该列显示变量的当前值。在调试程序时，可通过该列修改对应变量的值。
- ❑ 类型：显示每个声明变量类型。

在【本地窗口】右上角有个三个点的按钮，单击该按钮将打开【调用堆栈】对话框，如图 A-7 所示。

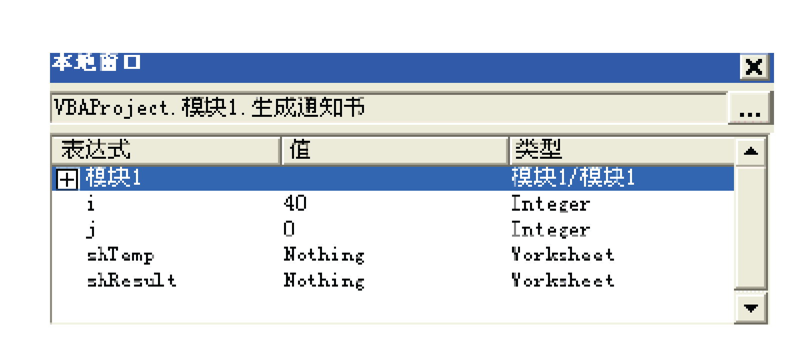


图 A-6 【本地窗口】列表框



图 A-7　调用堆栈

在【调用堆栈】窗口中显示所有调用的活动过程列表。使用该对话框调试嵌套程序时

在图 A-4 所示的【监视窗口】列表框中单击选中一个监视表达式，按 Delete 键可将该监视表达式清除。

2．使用快速监视

在没有定义监视表达式时，如果要查看一个表达式的值，可以使用快速监视。具体操作方法如下：

（1）在中断模式下，将光标放在变量名或表达式内部。

（2）单击菜单【调试】|【快速监视】命令（或按 Shift+F9 组合键），将打开如图 A-5 所示的【快速监视】对话框。

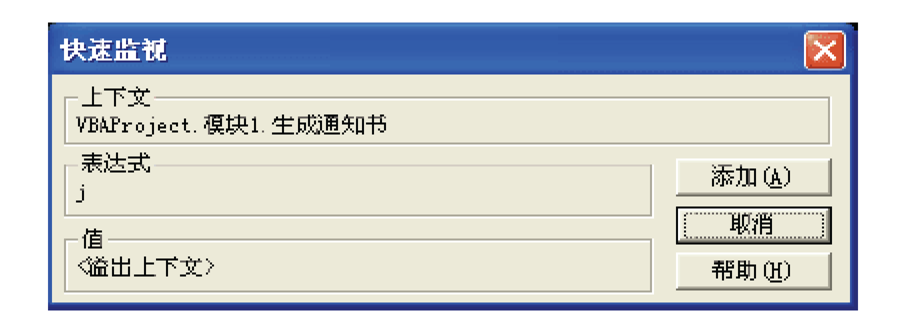


图 A-5 【快速监视】对话框

（3）单击【添加】按钮，可将表达式添加到【监视窗口】列表框中。

A.5　使用本地窗口

【本地窗口】列表框显示当前过程范围内所有变量的值。当执行由一个过程切换到另一过程时，【本地窗口】列表框的内容将转变为仅反应适用于当前过程的变量。

【本地窗口】列表框如图 A-6 所示，共包括下述 3 列。

- ❑ 表达式：显示当前过程里声明的变量名称。第一行显示的是“模块”变量，单击前面的加号，就可以查看是否定义有模块级变量。

注意：全局变量不会在“本地”窗口中显示。

- ❑ 值：在该列显示变量的当前值。在调试程序时，可通过该列修改对应变量的值。
- ❑ 类型：显示每个声明变量类型。

在【本地窗口】右上角有个三个点的按钮，单击该按钮将打开【调用堆栈】对话框，如图 A-7 所示。

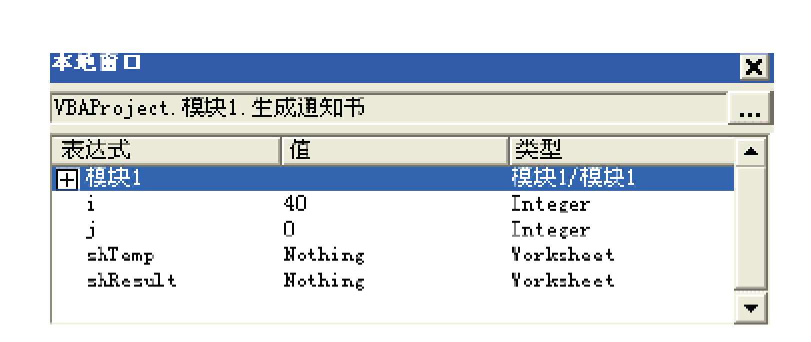


图 A-6 【本地窗口】列表框



图 A-7　调用堆栈

在【调用堆栈】窗口中显示所有调用的活动过程列表。使用该对话框调试嵌套程序时

特别有用，可将各过程之间的相互调用的关系显示出来。

A.6 使用立即窗口

【立即窗口】列表框就像一张草稿纸，可以用它来测试程序语句，或了解变量的值。可使用下列两种方法在【立即窗口】列表框中显示信息：

- 在应用程序代码中加入 Debug.Print 语句：可在【立即窗口】列表框输出内容；
- 在中断模式下：在【立即窗口】列表框中输入使用 Print 方法的语句来输出内容。

进入中断模式后，可把焦点移到【立即窗口】列表框中，输入 Print 方法，后面跟上任何有效的表达式，包括属性的表达式，然后按 Enter 键将在下方显示出运算的结果。

技巧：在【立即窗口】列表框中，可输入“？”（问号）来代替 Print 方法。

特别有用，可将各过程之间的相互调用的关系显示出来。

A.6　使用立即窗口

【立即窗口】列表框就像一张草稿纸，可以用它来测试程序语句，或了解变量的值。可使用下列两种方法在【立即窗口】列表框中显示信息：

- 在应用程序代码中加入 Debug.Print 语句：可在【立即窗口】列表框输出内容；
- 在中断模式下：在【立即窗口】列表框中输入使用 Print 方法的语句来输出内容。

进入中断模式后，可把焦点移到【立即窗口】列表框中，输入 Print 方法，后面跟上任何有效的表达式，包括属性的表达式，然后按 Enter 键将在下方显示出运算的结果。

技巧：在【立即窗口】列表框中，可输入“?”（问号）来代替 Print 方法。

附录 B　ASCII 码表

为了统一文字符号的编码标准，让不同厂牌机型的计算机都能使用同一套标准化的信息交换码，美国国家标准局特别制定了 ASCII 码（America Standard Code for Information Interchange，美国信息交换标准码），作为数据传输的标准码。早期使用 7 个位来表示英文字母、数字 0～9 及其他符号，现在则使用 8 个位，共可表示 256 个不同的文字与符号。

在 VBA 程序中，使用函数 ASC 可获得字母对应的 ASCII 码；使用函数 CHR 可将 ASCII 码转换为对应的字母。

ASCII码	控制字符	ASCII码	字符	ASCII码	字符	ASCII码	字符
000	NUL	032	(space)	064	@	096	`
001	SOH	033	!	065	A	097	a
002	STX	034	"	066	B	098	b
003	ETX	035	#	067	C	099	c
004	EOT	036	$	068	D	100	d
005	END	037	%	069	E	101	e
006	ACK	038	&	070	F	102	f
007	BEL	039	’	071	G	103	g
008	BS	040	(	072	H	104	h
009	HT	041	)	073	I	105	i
010	LF	042	*	074	J	106	j
011	VT	043	＋	075	K	107	k
012	FF	044	,	076	L	108	l
013	CR	045	－	077	M	109	m
014	SO	046	。	078	N	110	n
015	SI	047	/	079	O	111	o
016	DLE	048	0	080	P	112	p
017	DC1	049	1	081	Q	113	q
018	DC2	050	2	082	R	114	r
019	DC3	051	3	083	S	115	s
020	DC4	052	4	084	T	116	t
021	NAK	053	5	085	U	117	u
022	SYN	054	6	086	V	118	v
023	ETB	055	7	087	W	119	w
024	CAN	056	8	088	X	120	x
025	EM	057	9	089	Y	121	y
026	SUB	058	:	090	Z	122	z
027	ESC	059	;	091	[	123	{
028	FS	060	<	092	\	124	|
029	GS	061	=	093	]	125	}
030	RS	062	>	094	^	126	~
031	US	063	?	095	_	127	□